Fall 2016

Modified Hybrid Modeling Technique for Flexible Spin-Stabilized Spacecraft Applied to NASA's Magnetospheric MultiScale (MMS) Mission TableSat Generation IC (TableSat IC)

Chris Hashem
University of New Hampshire, Durham

MODIFIED HYBRID MODELING TECHNIQUE FOR FLEXIBLE SPIN-STABILIZED SPACECRAFT APPLIED TO NASA'S MAGNETOSPHERIC MULTISCALE (MMS) MISSION TABLESAT GENERATION IC (TABLESAT IC)

BY

CHRISTOPHER P. HASHEM
B.S. in Electromechanical Engineering, Wentworth Institute of Technology, Boston, 2013

THESIS

Submitted to the University of New Hampshire
In Partial Fulfillment of the Requirements for the Degree of
Master of Science
in
Mechanical Engineering

September, 2016

This thesis has been examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering by:

Thesis Director, Dr. May-Win L. Thein, Associate Professor of Mechanical Engineering and Ocean Engineering

Dr. Igor Tsukrov, Professor of Mechanical Engineering

Dr. Se Young Yoon, Assistant Professor of Electrical and Computer Engineering

On August 29, 2016

Original approval signatures are on file with the University of New Hampshire Graduate School.

Dedication

Dedicated to my father, for inspiring my love for math and the sciences. I know he would be proud to call his son a Master of Science.

Acknowledgments

Thank you to Prof. May-Win Thein for advising this project since January 2014 and giving me many opportunities during my time at UNH. Thank you to Prof. Igor Tsukrov and Prof. Pablo Yoon for serving on the thesis committee.

This research was sponsored by the NASA Goddard Space Flight Center MMS Grant. Special thanks go to Josephine San, Sam Placanica, and Juan Raymond of the MMS Attitude Control Systems Group for their collaborative efforts.

Huge shout out to Tom Fuller who not only helped me immensely on this work, but also has been a friend and colleague since the Wentworth days.

Big thanks to Mike Johnson and Shuai Chen for all their help and patience in the lab.

Thank you to the Department of Mechanical Engineering for supporting my studies at UNH through assistantship.

A huge thank you to Tracey Harvey and Lauren Foxall for being so pleasant all the time and making life easier for everyone in Mechanical Engineering.

Thanks to all my friends for believing in me and a very special thank you to my girlfriend, Alex, who never hesitated to try to make my life easier these past few months.

None of this would have been possible if not for the support from my mother, Rita, and my brother, Patrick. Thanks for always having my back and believing in me. Words can't express my appreciation of your unyielding support.

And of course, thanks to my homie Rex for never failing to put a smile on my face. Woof.

Contents

List of Figures

List of Tables

ABSTRACT

MODIFIED HYBRID MODELING TECHNIQUE FOR FLEXIBLE SPIN-STABILIZED SPACECRAFT APPLIED TO NASA'S MAGNETOSPHERIC MULTISCALE (MMS) MISSION TABLESAT GENERATION IC (TABLESAT IC)

by

Christopher P. Hashem

University of New Hampshire, September 2016

This continued research uses a hybrid dynamic algorithm to mathematically model the attitude dynamics of a flexible spacecraft. While a full finite element model would be the most accurate model, it also would be a huge computational burden on MMS. This method is meant to be an accurate approximation while still being computationally reasonable for on-board, real-time calculations. The algorithm uses Euler's Moment Equations to propagate the dynamics of the spacecraft hub while finite element analysis is used to calculate the flexible boom displacements and update the systems mass moment of inertia tensor. This algorithm is iterative and the FEM runs a user-defined number of times within each iteration. Experimental results show that the algorithm is reasonably accurate and can provide a viable approximation for flexible, spin-stabilized spacecraft. Lastly, comparison of controller performance shows Sliding Mode Control to be the most expensive yet most robust control method for this application.

Chapter 1

Introduction

1.1 Mission Background

The Mission

NASA's Magnetospheric MultiScale (MMS) Mission (launched in March 2015) was designed to study the interaction between Earth's magnetic field (the magnetosphere) and the Sun's solar winds [1]. As the solar winds bombard the magnetosphere, the plasma from the solar winds can disconnect from its magnetic field and reconnect with the magnetic field of Earth. The phenomenon, called magnetic field reconnection, is little-understood and occur in specific regions. These regions are highlighted in red in Fig. 1.1. Reconnection releases extremely large amounts of energy that can accelerate individual particles to near the speed of light and initiate large-scale flow of particles. The effects of these particles may be benign such as inducing the aurora we see near the poles. However, these particles can also interfere with GPS and telecommunications satellites in orbit and with nuclear fusion reactors, making this phenomenon an important area of study.

The Spacecraft

The MMS Mission features four spin-stabilized spacecraft (s/c) that fly in a tetrahedral formation, allowing for the four-dimensional (three spatial and one temporal) capture of magnetic and electric field data [1]. Each s/c, shown in Fig. 1.2 features an octagonal hub 3.4 meters in radius and 1.2 meters tall with a mass of 1,360 kilograms. Each s/c is also spin-stabilized and rotates axially at a rate of three revolutions per minute. After deployment, a total of eight booms extend from the s/c in the axial and spin planes. Of these eight booms, four are known as the Spin-plane Double Probes (SDP) and are the major concern of this research. These four booms are 60 meters in length and 1.6 millimeters in diameter with electric field sensors at the ends, making them very flexible. This fact coupled with the spin-stabilization means that the booms will be under constant force and deformation, resulting in a system with a time-varying second moment of mass, or mass moment of inertia.

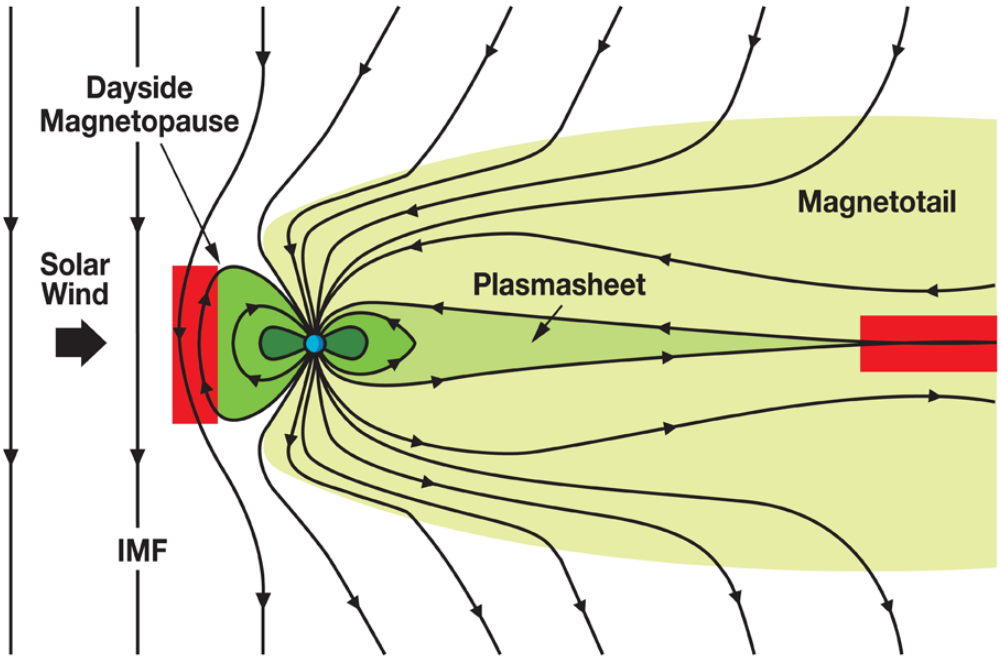

Figure 1.1: Solar winds interacting with Earth's magnetic field. Regions of magnetic field reconnection are highlighted in red.

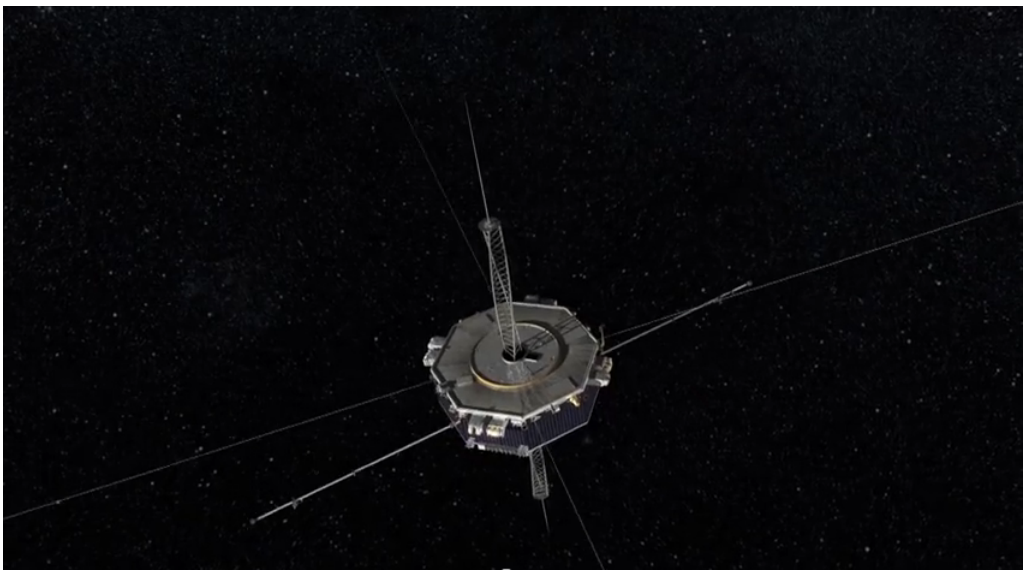

Figure 1.2: One of the four MMS s/c, with all booms deployed. The four spin-plane booms extend far off the screen [2].

Problem Statement

The time-varying characteristic of the mass moment of inertia is not a property that can be ignored. Because mass moment of inertia is a function of mass and distance from the center of rotation and because the booms are of significant length and have a lumped mass (sensor) at their tips, the deformation of the booms will have a significant effect on the mass moment of inertia of the system. This will, in turn, affect the ro-

tation rate for spin-stabilization, any attitude maneuvers, and any orbital maneuvers. This means that the mass moment of inertia will have to be modeled and updated over time. However, there is no quintessential convention for modeling a system with both flexible and rigid structures that does not require excessive computation. Rigid body dynamics cannot account for the large, nonlinear deformations of flexible structures and using strictly flexible structure dynamics would be impractical from a computational standpoint. It is important to have a accurate mathematical model for the s/c for control and monitoring purposes. As mentioned earlier, the four s/c must fly in a an exact tetrahedral formation and make various orbital maneuvers while maintaining formation and desired attitude.

1.2 Modeling Techniques

There have been other published methods that attempt to find a solution for this problem, all of which have their advantages, disadvantages, and impracticalities.

Rigid Body Dynamics

The easiest method is to neglect the dynamics of the flexible structures if their dynamics have an insignificant effect on the overall system dynamics. Modeling the system using this method would assume that the booms experience negligible deformations, meaning the mass moment of inertia of the system would be constant time. As mentioned in the previous section, this is not the case for MMS and a model like this would yield inaccurate results, voiding it from being a practical modeling option.

Finite Element Analysis

Another method is to use finite element analysis to propagate the dynamics of the entire system. This method is ideal when the computational load is not an issue. However, a finite element model (FEM) for such a large system can become very complex very quickly. As the reader will see in Chapter 2, even a finite element model as simple as the boom divided into two space frame elements has 18 degrees of freedom (DOFs). An FEM with a finer mesh for the entire rigid body and booms would be cumbersome without a dedicated finite element analysis software package. Then interfacing between softwares for finite element analysis and numerical computation becomes an issue and significantly slows down the speed of the simulation. For a large complex system such as MMS, using strictly finite element analysis to model the entire system would be impractical for on-board, real-time calculations.

Multibody Physics

In Stoneking [3], the NASA Goddard MMS Attitude Control System (ACS) group attempted to model the flexible booms as a series of rigid bodies connected by gimbaled joints.. The nonlinear equations of motion are derived from Euler's and Newton's laws of motion for two bodies connected by a spherical joint. Similarly, the model can be extrapolated to any number of bodies by concatenating the matrices that compose the equations of motion according to the connectivity of the bodies. The equations for a spherical joint can be manipulated to that of a gimbal joint, sacrificing certain

symmetries but creating new patterns and allowing for model assembly by inspection (one of the goals of the paper). Results show that this method achieved its goals and captured important dynamic features of the system while being relatively easy to implement.

Hybrid Algorithm

In Medas and Thein [4], a hybrid algorithm is introduced in which a combination of rigid body dynamics and finite element analysis is used to model the dynamics of the entire system. The output angular displacement from the rigid body equations is used to calculate the boundary conditions for the FEM. The output displacement vector from the FEM is used to update the mass moment of inertia for the rigid body equations. The boundary condition calculation and updating of the mass moment of inertia is what links the two methods (rigid body equations and FEM) together. Results from this method showed qualitatively correct behavior but yielded instabilities under certain conditions, particularly in the FEM. A reaction moment acting on the hub from the deflection of the boom is added to propagate boom motion to the hub. While this rectifies certain instabilities, it creates new ones in other circumstances.

Modified Hybrid Algorithm

This research is an expansion of Medas' algorithm that addresses some issues with the results obtained from Medas and Thein [4]. A different method of solving the boundary conditions for the FEM is investigated in which all nodes of the boom are placed at an initial displacement corresponding to the rotation of the hub during the previous time step. This method allows for the simplification of the FEM by inducing zero-elimination at the base node and uses displacement as the driver instead of a force. A different method of numerical integration within the FEM, the Newmark-β method, is implemented to achieve greater stability and is compared to the central difference method used in the original hybrid algorithm.

1.3 Thesis Contributions

The primary objective of this research is to create a self-contained, computationally efficient, proof-of-concept mathematical model for the MMS spacecraft by addressing the issues with the Medas and Thein hybrid algorithm. The secondary objective is to prove that this algorithm is accurate by experimentally validating the results with the experimental test bed MMS TableSat Generation 1C (TableSat 1C). Modifications are made to the test bed in order to obtain more reliable data than obtained by Medas and Thein. The tertiary objective is to apply various controllers, including a PID controller, Sliding Mode Controller (SMC), and a Linear Quadratic Regulator (LQR), to the system to investigate which controller performs best against disturbances to the system. Ultimately, the goal is to show that this modified hybrid algorithm is a viable, computationally less-intensive approximation to existing mathematical models. The NASA MMS s/c are used as the test s;c, without loss of generality. This hybrid method is also investigated as a viable option to predicts/c motion for general flexible spin-stabilized s/c, particularly during mission operations (science modes, orbital maneuvers, etc).

1.4 Thesis Outline

Chapter 1: Introduction This chapter introduces the mission background, problem statement, background research, thesis contributions, and the thesis outline.
Chapter 2: Modified Hybrid Algorithm This chapter discusses the motivations for altering parts of the original algorithm of Medas and Thein. The modified hybrid algorithm is presented in detail. The method for modeling the rigid body equations is Euler's Moment Equations and the method for modeling the flexible structures is finite element analysis. The Newmark-β integration method is used instead of the central difference method. These two methods are discussed separately, then their integration into one modeling algorithm is proposed.
Chapter 3: Controller Design Three controllers are designed for control of the s/c spin rate and presented in order to compare disturbance rejection and control effort efficiency during simulation. The control methods used are PID, SMC, and LQR.
Chapter 4: Experimental Platform and Verification The developed experimental platform is discussed along with experimental results and analy-sis on sensor accuracy. Points of interest are the measured rate of yaw rotation and boom deflections.
Chapter 5: Modified Algorithm Simulations Simulations are presented in order to determine the validity of the new algorithm. Tests performed include conditions of s/c spin-up from rest to steady state, results of steady state disturbance, free response dynamics as a result of input deflection on boom, and conditions during true MMS.
Chapter 6: Discussion of Results This section discusses the results of the theoretical derivations, the simulations, and the experiments presented in the previous chapters. This also section discusses present successes and failures and compares the various methods of modeling and performance of controllers.
Chapter 7: Conclusions The thesis conclusions are made, relating the objectives set forth in Chapter 1 to the progress that was made throughout the thesis. Suggestions for future work that could improve and further generalize the modified algorithm are discussed in this chapter as well.

Chapter 2

Modified Hybrid Algorithm

In this chapter, the details of the hybrid modeling algorithm are explained in detail. Each section of the algorithm is explained independently as well as the mathematical relationships that interface the different sections together. A flowchart of the algorithm is shown in Fig. 2.1. Certain parameters of the simulation must be defined before the

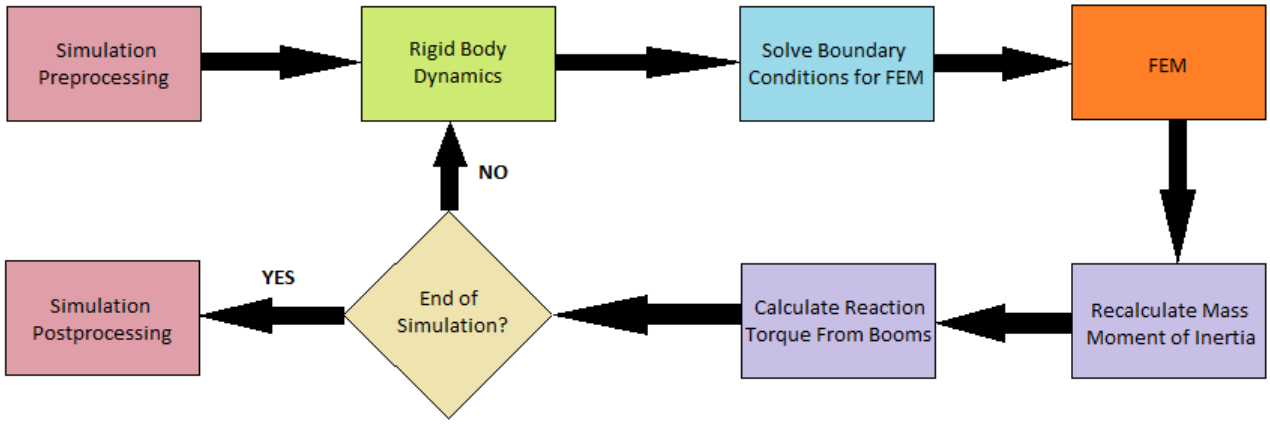

Figure 2.1: Flowchart of the modified hybrid algorithm

simulation can run such as the desired rates of rotation, the time step, the material and geometric properties of both the s/c hub and the booms, the initial conditions of the s/c hub and the booms, and the FEM mass, spring, and damping matrices. Formulation of the FEM matrices will be discussed later in this chapter. The algorithm is an iterative process with each iteration occurring over the course of a time step. This time step must be small enough to ensure stability and avoid aliasing, though cannot be so small as to become computationally impractical. For the purpose of this research, the author defined the time step to be 0.01 seconds. At the beginning of each time step, the rigid body dynamics are propagated according to the desired rotation rates. The boundary conditions for the FEM are calculated from the output of the rigid body dynamics before the start of the finite element analysis. The FEM has a much smaller time step than the algorithm (for the sake of differentiation, the FEM time step will be referred to as Δt). For the reader's reference, Δt is one hundred times smaller than the algorithm time step for the sake of this research, though the criteria for calculating Δt will also be discussed later in this chapter. Therefore, the FEM runs through one hundred iterations for every one iteration of the algorithm. Once the finite element analysis has finished running, the last value of the displacement vector

is used to update the mass moment of inertia and calculate the reaction moment from the deflection of the booms. These two values are then used in the next iteration of the rigid body dynamics, as shown in Fig. 2.1, and the loop continues.

It is important to note that there are five total coordinate systems used in this algorithm. One coordinate system has its origin at the center of the s/c hub, with the xy-plane coinciding with the spin and the z-axis orthogonal to the spin-plane. The hybrid algorithm uses this coordinate system as its reference. The other four

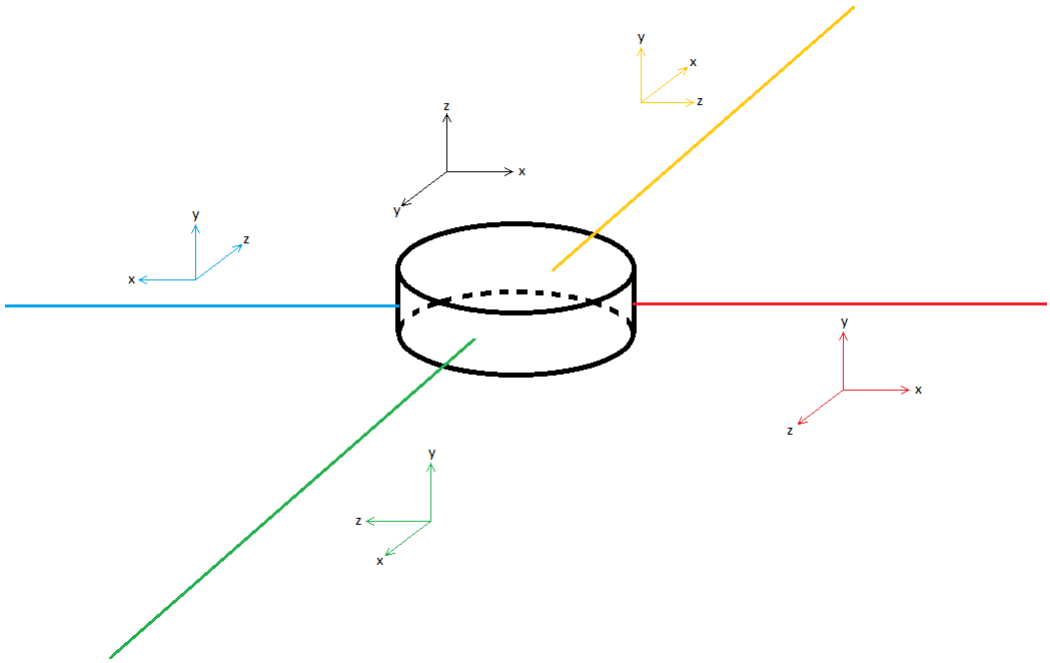

Figure 2.2: Visualization of the alignment of the five coordinate systems

coordinate systems are used in the FEM, with the origins at the base of each node, the x-axis along the length of the boom, the z-axis in the spin-plane, and the y-axis orthogonal to the spin-plane. A depiction of the coordinate system is shown in Fig. 2.2

2.1 Rigid Body Dynamics

In order to model the central hub of the MMS spacecraft, Euler's Moment Equations, shown in Eq. (2.1), are used [5].

$$\dot{\omega}_x = \frac{1}{I_x}[M_x - (I_z - I_y)\omega_y\omega_z] \tag{2.1a}$$

$$\dot{\omega}_y = \frac{1}{I_y}[M_y - (I_x - I_z)\omega_z\omega_x] \tag{2.1b}$$

$$\dot{\omega}_z = \frac{1}{I_z}[M_z - (I_y - I_x)\omega_x\omega_y] \tag{2.1c}$$

In these equations,

- ω_x, ω_y, ω_z are the rates of rotation, about their respective axes, of the s/c hub
- M_x, M_y, M_z are the external moments, about their respective axes, applied on the s/c hub from the thrusters or disturbances

- I_x, I_y, I_z are the principal second mass moments, about their respective axes, of the s/c
- the dot notation represents a time derivative

These nonlinear equations are modeled using a numerical simulation software. Once the value for $\dot{\omega}$ is obtained, it is integrated twice with respect to time in order to obtain the rates of rotation and the angular positions, respectively, about all three axes. The rotation rates are used as sensor feedback for the rate control and the angular position is used to set up the FEM.

The simulation model uses a fixed time step solver where the length of the time step is as defined previously in this chapter. Keeping in mind that the simulation model runs once at the beginning of each algorithm iteration, the simulation software length (not to be confused with the general length of the simulation) is also one fixed time step. Because each iteration of the simulation software model is not continuous with the last, steps must be taken so that continuous-time integration and differentiation can be used. When a quantity is integrated, the product is stored in the memory and becomes the initial condition of the integration on the next iteration, preserving the cumulative summation of the quantity. For differentiation, the quantity at the last time step is subtracted from the quantity at the current time step. The difference is then divided by the time step size. The top level of the simulation model is shown in Fig. 2.3.

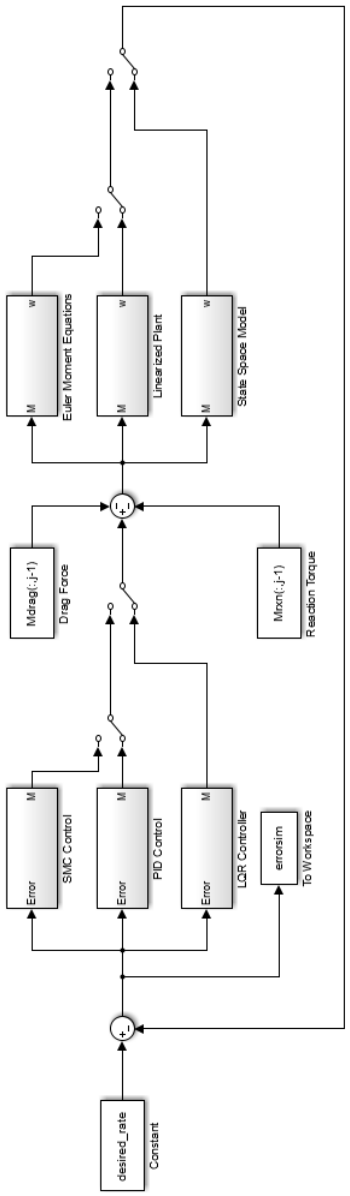

Figure 2.3: Top level of the simulation model representing the rigid body dynamics

2.2 Finite Element Model

2.2.1 Finite Element Type, Geometry, and Global Displacement Vector Construction

Developing a finite element model begins with deciding which element type suits the requirements of the system. For the flexible booms, a finite element that can bend in two directions and displace axially is needed. The space frame element suits these requirements, shown in Fig. 2.4, features twelve total DOFs. One displacement DOF and one bending DOF in each direction at each node. The displacement vector for the space frame element in the local frame (denoted by the prime notation) is shown in

Eq. (2.2) [7].

$$\vec{U}'_{[12x1]} = \begin{Bmatrix} u'_1 \\ v'_1 \\ w'_1 \\ \theta'_{x1} \\ \theta'_{y1} \\ \theta'_{z1} \\ u'_2 \\ v'_2 \\ w'_2 \\ \theta'_{x2} \\ \theta'_{y2} \\ \theta'_{z2} \end{Bmatrix} \tag{2.2}$$

The MMS booms are not expected to experience torsion due to the assumption that the geometry of the boom is that of a slender rod [4]. Thus, the θ_{x1} and θ_{x2} DOFs can be assumed to be zero for all Δt and therefore eliminated from the displacement vector. A cantilevered two element model is used in this FEM, as depicted in Fig. 2.5, and the connectivity table for such a geometry is shown in Table 2.1. Using the connectivity

Element	Node $\hat{\mathbf{1}}$	Node $\hat{\mathbf{2}}$
1	Node 1	Node 2
2	Node 2	Node 3

Table 2.1: Connectivity table for two element cantilevered system

table, we concatenate Eq. (2.2) to build a two-element displacement vector. Keeping

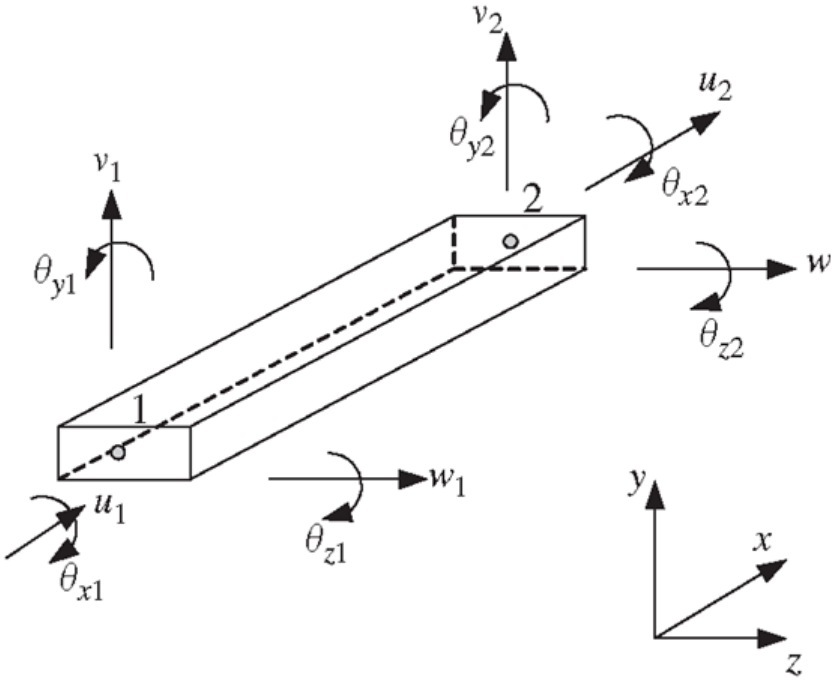

Figure 2.4: Space frame element with all degrees of freedom labeled [6]

in mind to neglect torsion, we obtain our global displacement vector, Eq. (2.3).

$$\vec{U}_{[15x1]} = \begin{Bmatrix} u'_1 \\ v'_1 \\ w'_1 \\ \theta'_{y1} \\ \theta'_{z1} \\ u'_2 + u'_1 \\ v'_2 + v'_2 \\ w'_2 + w'_1 \\ \theta'_{y2} + \theta'_{y1} \\ \theta'_{z2} + \theta'_{z1} \\ u'_2 \\ v'_2 \\ w'_2 \\ \theta'_{y2} \\ \theta'_{z2} \end{Bmatrix} = \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ \theta_{y1} \\ \theta_{z1} \\ u_2 \\ v_2 \\ w_2 \\ \theta_{y2} \\ \theta_{z2} \\ u_3 \\ v_3 \\ w_3 \\ \theta_{y3} \\ \theta_{z3} \end{Bmatrix} \tag{2.3}$$

2.2.2 Global Mass, Stiffness, and Damping Matrix Construction

Assuming a linear elastic response, the local consistent mass and stiffness matrices are shown in Eq. (2.4) and Eq. (2.5), respectively [7].

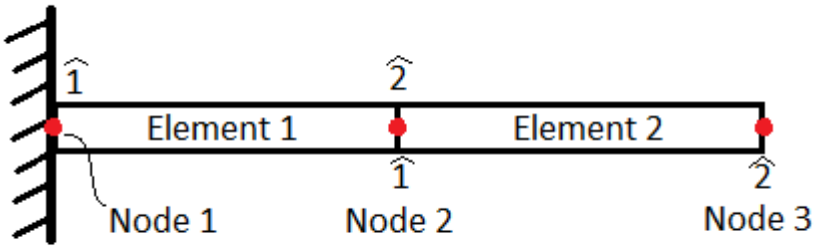

Figure 2.5: Geometry of cantilevered two-element FEM. Hat denotes local element node and no hat represents global nodes

$$\vec{M}'_{[10x10]} = lA\rho \begin{bmatrix} 70 & 0 & 0 & 0 & 0 & 35 & 0 & 0 & 0 & 0 \\ 0 & 78 & 0 & 0 & 11 & 0 & 27 & 0 & 0 & -6.5l \\ 0 & 0 & 78 & -11l & 0 & 0 & 0 & 27 & 6.5l & 0 \\ 0 & 0 & -11l & 2l^2 & 0 & 0 & 0 & -6.5l & -1.5l^2 & 0 \\ 0 & 11l & 0 & 0 & 2l^2 & 0 & 6.5l & 0 & 0 & -1.5l^2 \\ 35 & 0 & 0 & 0 & 0 & 70 & 0 & 0 & 0 & 0 \\ 0 & 27 & 0 & 0 & 6.5l & 0 & 78 & 0 & 0 & -11l \\ 0 & 0 & 27 & -6.5l & 0 & 0 & 0 & 78 & 11l & 0 \\ 0 & 0 & 6.5l & -1.5l^2 & 0 & 0 & 0 & 11l & 2l^2 & 0 \\ 0 & -6.5l & 0 & 0 & -1.5l^2 & 0 & -11l & 0 & 0 & 2l^2 \end{bmatrix} = lA\rho \begin{bmatrix} M_a & M_b \\ M_c & M_d \end{bmatrix} \tag{2.4}$$

$$\vec{K}'_{[10x10]} = \begin{bmatrix} \frac{EA}{l} & 0 & 0 & 0 & 0 & \frac{-EA}{l} & 0 & 0 & 0 & 0 \\ 0 & \frac{12E\hat{I}_{zz}}{l^3} & 0 & 0 & \frac{6E\hat{I}_{zz}}{l^2} & 0 & \frac{-12E\hat{I}_{zz}}{l^3} & 0 & 0 & \frac{6EI_{zz}}{l^2} \\ 0 & 0 & 12EI_{yy}/l^3 & -6EI_{yy}/l^2 & 0 & 0 & 0 & -12EI_{yy}/l^3 & -6EI_{yy}/l^2 & 0 \\ 0 & 0 & -6EI_{yy}/l^2 & 4EI_{yy}/l & 0 & 0 & 0 & 6EI_{yy}/l^2 & 2EI_{yy}/l & 0 \\ 0 & 6EI_{zz}/l^2 & 0 & 0 & 4EI_{zz}/l & 0 & -6EI_{zz}/l^2 & 0 & 0 & 2EI_{zz}/l \\ -EA/l & 0 & 0 & 0 & 0 & EA/l & 0 & 0 & 0 & 0 \\ 0 & -12EI_{zz}/l^3 & 0 & 0 & -6EI_{zz}/l^2 & 0 & 12EI_{zz}/l^3 & 0 & 0 & -6EI_{zz}/l^2 \\ 0 & 0 & -12EI_{yy}/l^3 & 6EI_{yy}/l^2 & 0 & 0 & 0 & 12EI_{yy}/l^3 & 6EI_{yy}/l^2 & 0 \\ 0 & 0 & -6EI_{yy}/l^2 & 2EI_{yy}/l & 0 & 0 & 0 & 6EI_{yy}/l^2 & 4EI_{yy}/l & 0 \\ 0 & 6EI_{zz}/l^2 & 0 & 0 & 2EI_{zz}/l & 0 & -6EI_{zz}/l^2 & 0 & 0 & 4EI_{zz}/l \end{bmatrix} \tag{2.5}$$

$$\vec{K'}_{[10x10]} = \begin{bmatrix} K_a & K_b \\ K_c & K_d \end{bmatrix} \tag{2.6}$$

In these matrices,

- l is the element length, $l = 0.2149m$
- ρ is the material density, $\rho = 8000\,{}^{\mathrm{kg}}/_{\mathrm{m}^3}$
- A is the cross-sectional area, $A = 9.5806 \times 10^{-7} m^2$
- E is the modulus of elasticity, $E = 200 \times 10^9 Pa$
- $\hat{I}_{yy}$ and $\hat{I}_{zz}$ are the second moments of area about their respective axes (not to be confused with the second moment of mass)

The damping matrix is found by evaluating Eq. (2.7) [7],

$$\vec{B'} = \mu \int\int\int \vec{N}^T \vec{N} dV \tag{2.7}$$

where

- μ is the damping coefficient, $\mu = 0.0409\,{}^{\mathrm{N{\cdot}s}}/_{\mathrm{m}}$
- N is the shape function in Eq. (2.8) [7]

$$\vec{N}^T_{[10x1]} = \begin{bmatrix} 1 - \dfrac{x}{l} \\ \dfrac{2x^3 - 3lx^2 + l^3}{l^3} \\ \dfrac{2x^3 - 3lx^2 + l^3}{l^3} \\ \dfrac{lx^3 - 2l^2x^2 + l^3x}{l^3} \\ \dfrac{lx^3 - 2l^2x^2 + l^3x}{l^3} \\ \dfrac{x}{l} \\ \dfrac{-2x^3 + 3lx^2}{l^3} \\ \dfrac{-2x^3 + 3lx^2}{l^3} \\ \dfrac{lx^3 - l^2x^2}{l^3} \\ \dfrac{lx^3 - l^2x^2}{l^3} \end{bmatrix} \tag{2.8}$$

The damping coefficient μ is found experimentally by deflecting the boom and using the Log Decrement Method to analyze the resulting free motion [4]. The resulting damping matrix is shown in Eq. (2.9).

$$\vec{B}'_{[10x10]} = \begin{bmatrix} 293.0197 & 307.6707 & 307.6707 & 9.4468 & 9.4468 & 146.5099 & 131.8589 & 131.8589 & -6.2978 & -6.2978 \\ 307.6707 & 326.5077 & 326.5077 & 9.8966 & 9.8966 & 131.8589 & 113.0219 & 113.0219 & -5.8480 & -5.8480 \\ 307.6707 & 326.5077 & 326.5077 & 9.8966 & 9.8966 & 131.8589 & 113.0219 & 113.0219 & -5.8480 & -5.8480 \\ 9.4468 & 9.8966 & 9.8966 & 0.3867 & 0.3867 & 6.2978 & 5.8480 & 5.8480 & -0.2901 & -0.2901 \\ 9.4468 & 9.8966 & 9.8966 & 0.3867 & 0.3867 & 6.2978 & 5.8480 & 5.8480 & -0.2901 & -0.2901 \\ 146.5099 & 131.8589 & 131.8589 & 6.2978 & 6.2978 & 293.0197 & 307.6707 & 307.6707 & -9.4468 & -9.4468 \\ 131.8589 & 113.0219 & 113.0219 & 5.8480 & 5.8480 & 307.6707 & 326.5077 & 326.5077 & -9.8966 & -9.8966 \\ 131.8589 & 113.0219 & 113.0219 & 5.8480 & 5.8480 & 307.6707 & 326.5077 & 326.5077 & -9.8966 & -9.8966 \\ -6.2978 & -5.8480 & -5.8480 & -0.2901 & -0.2901 & -9.4468 & -9.8966 & -9.8966 & 0.3867 & 0.3867 \\ -6.2978 & -5.8480 & -5.8480 & -0.2901 & -0.2901 & -9.4468 & -9.8966 & -9.8966 & 0.3867 & 0.3867 \end{bmatrix} \times 10^{-5} \tag{2.9}$$

$$\vec{B}'_{[10x10]} = \begin{bmatrix} B_a & B_b \\ B_c & B_d \end{bmatrix} \tag{2.10}$$

Much like the displacement vector, the global mass, stiffness, and damping matrices are found by concatenating the local matrices according to Table 2.1. The concatenations are shown in Eq. (2.11), Eq. (2.12), and Eq. (2.13) and the resulting mass, stiffness, and damping matrices are shown in Appendix A.

$$\vec{M}_{[15x15]} = \begin{bmatrix} M_a & M_b & 0 \\ M_c & M_d + M_a & M_b \\ 0 & M_b & M_d \end{bmatrix} \tag{2.11}$$

$$\vec{K}_{[15x15]} = \begin{bmatrix} K_a & K_b & 0 \\ K_c & K_d + K_a & K_b \\ 0 & K_b & K_d \end{bmatrix} \tag{2.12}$$

$$\vec{B}_{[15x15]} = \begin{bmatrix} B_a & B_b & 0 \\ B_c & B_d + B_a & B_b \\ 0 & B_b & B_d \end{bmatrix} \tag{2.13}$$

2.2.3 Methods of Numerical Integration in Time

Two methods of numerical integration were considered. Like the original hybrid algorithm of Medas and Thein, the Central Difference Method is used. The Central Difference Method is found to yield marginally stable results. As a result, the Newmark-β method is investigated and found to yield more stable results. Both methods are explained in detail in the following sections.

The Central Difference Method

The Central Difference Method uses the current time step and last time step of the displacement vector to calculate the displacement vector at the next time step, as depicted in Fig. 2.6. From this concept, Eq. (2.14) and Eq. (2.15) can be derived and

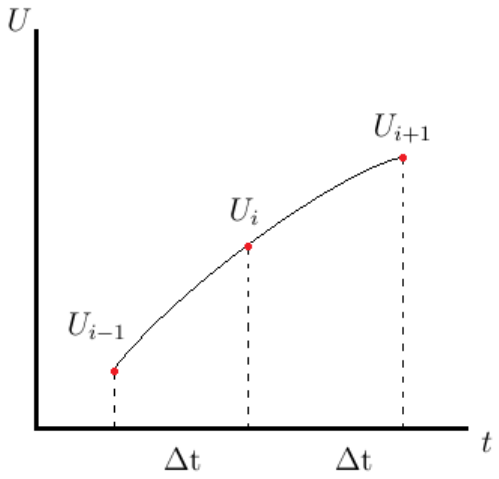

Figure 2.6: Arbitrary plot of displacement over time.

substituted into the time derivative of the displacement vector $\vec{U}_i$ [7], such that

$$\dot{\vec{U}}_i = \frac{\vec{U}_{i+1} - \vec{U}_{i-1}}{2\Delta t} \tag{2.14}$$

$$\ddot{\vec{U}}_i = \frac{\vec{U}_{i+1} - 2\vec{U}_i - \vec{U}_{i-1}}{\Delta t^2} \tag{2.15}$$

Using Eq. (2.16), the relation for Eq. (2.17) can be obtained.

$$\vec{M}\ddot{\vec{U}} = \vec{F} - \vec{B}\dot{\vec{U}} - \vec{K}\vec{U} \tag{2.16}$$

$$\vec{M}\vec{U}_{i+1} = \vec{F}_i\Delta t^2 + (2\vec{M} - \vec{K}\Delta t^2)\vec{U}_i - \vec{M}\vec{U}_{i-1} \tag{2.17}$$

In order to introduce damping into the equation, the effective mass matrix, Eq. (2.18), must be constructed such that,

$$\vec{M}_{eff} = \frac{\vec{M}}{\Delta t^2} + \frac{\vec{B}}{2\Delta t} \tag{2.18}$$

The effective mass matrix is then substituted into Eq. (2.17), resulting in Eq. (2.19).

$$\vec{M}_{eff}\vec{U}_{i+1} = \vec{F}_i\Delta t^2 + (2\vec{M} - \vec{K}\Delta t^2)\vec{U}_i - \vec{M}_{eff}\vec{U}_{i-1} \tag{2.19}$$

As mentioned previously, there is a specific criteria for selecting Δt. The central difference method is conditionally stable based on Δt. In order to guarantee stability, Δt must meet the following requirement:

$$\Delta t < t_{crit} = \frac{2}{\omega_{n,max}} \tag{2.20}$$

where $\omega_{n,max}$ si the system's largest natural frequency and is obtained from the following relation:

$$-\vec{M}\vec{\omega}_n^2 + \vec{K} = 0 \tag{2.21}$$

With this information, a suitable Δt can be selected to ensure stability of the FEM.

Because the Central Difference Method relies on past states, the imaginary value U_{-1} must be found for the first step of the simulation when t=0. This imaginary value can be found using

$$\vec{U}_{-1} = \vec{U}_0 - \Delta t\dot{\vec{U}}_0 + \frac{\Delta t^2}{2}\ddot{\vec{U}}_0 \tag{2.22}$$

Once all this information is available, the algorithm implementing the Central Difference Method is as follows:

- Initialization:
  1. Calculate $\ddot{\vec{U}}_0$ from $\vec{M}\ddot{\vec{U}}_0 = \vec{F}_0 - \vec{K}\vec{U}_0 - \vec{B}\dot{\vec{U}}_0$
  2. Calculate $\vec{U}_{-1}$ from Eq. (2.22)
- Computed each time step:
  1. Obtain $\vec{F}_i$
  2. Calculate $\vec{U}_{i+1}$ from 2.19
  3. Calculate $\dot{\vec{U}}_i$ from 2.14
  4. Calculate $\ddot{\vec{U}}_i$ from 2.15

The Newmark-β Method

The Newmark-β Method uses finite difference integration to build relations between U_{i+1} and U_i and its time derivatives:

$$\vec{\ddot{U}}^* = \frac{\vec{\dot{U}}_{i+1} - \vec{\dot{U}}_i}{\Delta t} \tag{2.23}$$

$$\vec{\dot{U}}_{i+1} = \vec{\dot{U}}_i + \Delta t \vec{\ddot{U}}^* \tag{2.24}$$

$$\vec{U}_{i+1} = \vec{U}_i + \Delta t \vec{\dot{U}} + \frac{\Delta t^2}{2} \vec{\ddot{U}}^* \tag{2.25}$$

Eq. (2.24) is simply a manipulation of Eq. (2.23) and Eq. (2.25) is an integration of Eq. (2.24). This method also introduces two new parameters, γ and β, to improve accuracy and stability over the Central Difference Method [7]. For $\gamma = \frac{1}{2}$ and $\beta = 0$, the Newmark method is identical to the central difference method. For $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$, the Newmark method is stable for all values of Δt. These are the values used in the modified hybrid algorithm. For the sake of ease of notation, a series of coefficients are defined below:

$$a_0 = \frac{1}{\beta \Delta t^2} \tag{2.26a}$$

$$a_1 = \frac{\gamma}{\beta \Delta t} \tag{2.26b}$$

$$a_2 = \frac{1}{\beta \Delta t} \tag{2.26c}$$

$$a_3 = \frac{1}{2\beta} - 1 \tag{2.26d}$$

$$a_4 = \frac{\gamma}{\beta} - 1 \tag{2.26e}$$

$$a_5 = \left(\frac{\gamma}{\beta} - 2\right) \frac{\Delta t}{2} \tag{2.26f}$$

$$a_6 = (1 - \gamma)\Delta t \tag{2.26g}$$

$$a_7 = \gamma \Delta t \tag{2.26h}$$

With these definitions in Eq. (2.26), one can redefine the relationships between $\vec{U}$, $\vec{\dot{U}}$, and $\vec{\ddot{U}}$ to account for γ and β. $\vec{\ddot{U}}^*$ is defined such that

$$\vec{\ddot{U}}^* = (1 - \gamma)\vec{\ddot{U}}_i + \gamma \vec{\ddot{U}}_{i+1} \tag{2.27}$$

and is used with Eq. (2.24) to obtain

$$\vec{\dot{U}}_{i+1} = \vec{\dot{U}}_i + [(1 - \gamma)\vec{\ddot{U}}_i + \gamma \vec{\ddot{U}}_{i+1}]\Delta t \tag{2.28a}$$

$$\vec{\dot{U}}_{i+1} = \vec{\dot{U}}_i + a_6 \vec{\ddot{U}}_i + a_7 \vec{\ddot{U}}_{i+1} \tag{2.28b}$$

Then, $\vec{\ddot{U}}^*$ is defined as

$$\vec{\ddot{U}}^* = (1 - 2\beta)\vec{\ddot{U}}_i + 2\beta \vec{\ddot{U}}_{i+1} \tag{2.29}$$

and combined with Eq. (2.25), resulting in

$$\vec{U}_{i+1} = \vec{U}_i + \dot{\vec{U}}_i \Delta t + [(1-2\beta)\ddot{\vec{U}}_i + 2\beta\ddot{\vec{U}}_{i+1}] \tag{2.30}$$

which can, in turn, be represented as

$$\ddot{\vec{U}}_{i+1} = \frac{1}{\beta \Delta t}[\vec{U}_{i+1} - \vec{U}_i - \Delta t \dot{\vec{U}}_i - \Delta t^2(\frac{1}{2} - \beta)\ddot{\vec{U}}_i] \tag{2.31a}$$

$$\ddot{\vec{U}}_{i+1} = a_0(\vec{U}_{i+1} - \vec{U}_i) - a_2\dot{\vec{U}}_i - a_3\ddot{\vec{U}}_i \tag{2.31b}$$

Next, the effective stiffness matrix $\vec{K}_{eff}$ is defined

$$\vec{K}_{eff} = \vec{K} + a_0\vec{M} + a_1\vec{B} \tag{2.32}$$

and the effective force matrix $\vec{F}_{eff}$ as

$$\vec{F}_{eff,i+1} = \vec{F}_{i+1} + \vec{M}(a_0\vec{U}_i + a_2\dot{\vec{U}}_i + a_3\ddot{\vec{U}}_i) + \vec{B}(a_1\vec{U}_i + a_4\dot{\vec{U}}_i + a_5\ddot{\vec{U}}_i) \tag{2.33}$$

Finally, $\vec{U}_{i+1}$ is obtained

$$\vec{U}_{i+1} = \vec{K}_{eff}^{-1}\vec{F}_{eff,i+1} \tag{2.34}$$

The Newmark-β algorithm proceeds as follows:

- Initialization:
  1. Formulate $\vec{K}_{eff}$ using Eq. (2.32)
  2. Calculate $\ddot{\vec{U}}_0$ from $\vec{M}\ddot{\vec{U}}_0 = \vec{F}_0 - \vec{K}\vec{U}_0 - \vec{B}\dot{\vec{U}}_0$
- Computed each time step:
  1. Formulate $\vec{F}_{eff,i+1}$ using Eq. (2.33)
  2. Calculate $\vec{U}_{i+1}$ from Eq. (2.34)
  3. Calculate $\ddot{\vec{U}}_i$ from Eq. (2.31)
  4. Calculate $\dot{\vec{U}}_i$ from Eq. (2.28)

2.3 FEM Boundary Conditions

With the rigid body and flexible body methods of analysis described in detail, the methods by which they are linked into a hybrid algorithm are described in detail in the following sections. This section details the boundary conditions of the rigid body dynamics which, in turn, drive the FEM.

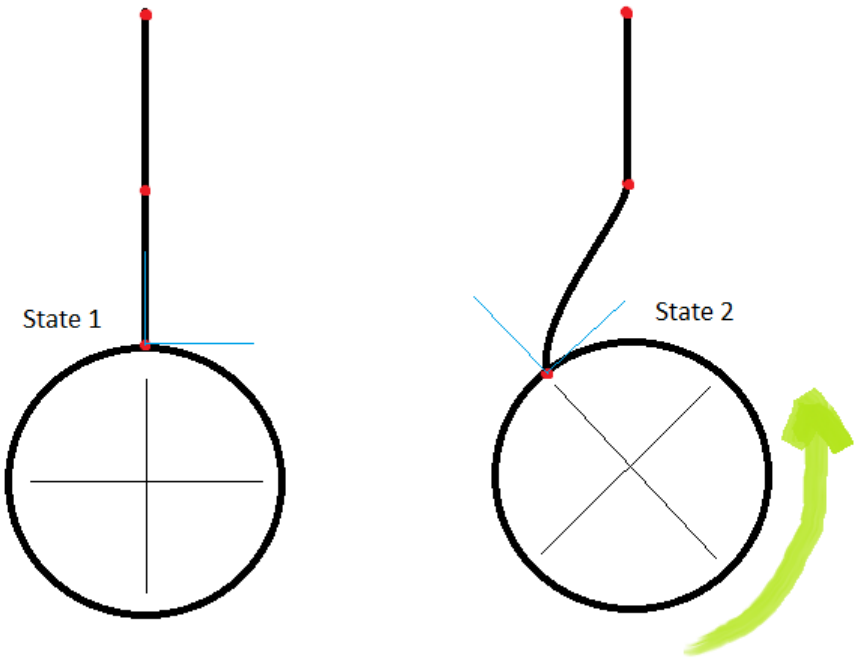

Figure 2.7: Depiction of rigid body rotation with one boom

2.3.1 Driving Mechanism

As mentioned before, the angular position obtained from Euler's Moment Equations is used to calculate the boundary conditions for the FEM. Take, for example, the visualization shown in Fig. 2.7. Let the rigid body rotate about its z-axis by some arbitrary angle. As the rigid body rotates, Node 1 (the base of the boom) and the coordinate system of the boom move with the rigid body. Node 2 and Node 3, however, remain in the same position as before the rotation. This creates a non-zero initial displacement which can be modeled in the FEM. Instead of a force being the driving mechanism for the FEM, an initial displacement is the driving mechanism for the free response of the boom. Though rotation about only one set of axes is shown here, the concept is the same for all four flexible booms. A coordinate transformation must be made in order to translate and rotate the boom's coordinate system according to the rotation of the rigid body. The procedure for doing so is detailed in the following section.

2.3.2 Coordinate Transformation

For a boom aligned with the x-axis of the rigid body, we refer to Fig. 2.8 which depicts the distances between where the zero-displacement points (in red) for Nodes 2 and 3 are positioned in the current time step and where the zero-displacement points are positioned in the past time step are depicted. Using the Small Angle Theorem, one can assume that these distances can be represented by chord lengths. The equation for the length of a chord is defined as

$$chord = 2R\,sin\left(\tfrac{1}{2}\theta\right) \tag{2.35}$$

and depicted in Fig. 2.9, where R is the distance from the center of the circle and ϕ is the angle of the corresponding arc length. For simplicity's sake, $chord_2$ and $chord_3$

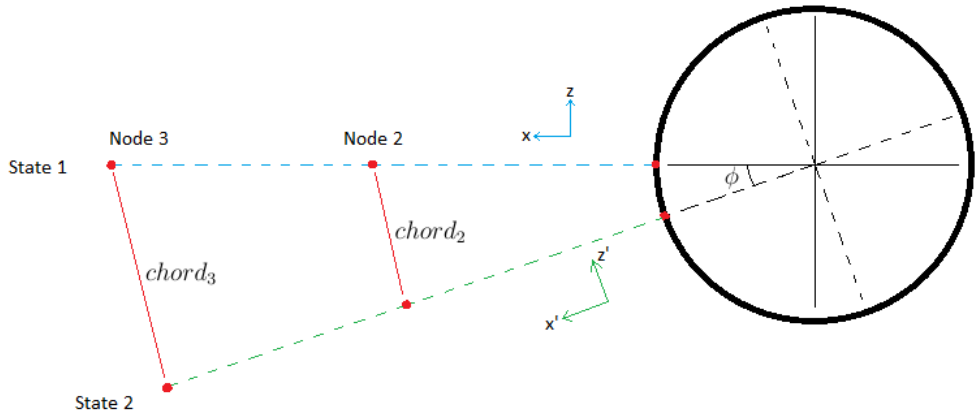

Figure 2.8: Depiction of the motion of the boom's coordinate system relative to the rigid body's rotation and the zero-displacement points (in red)

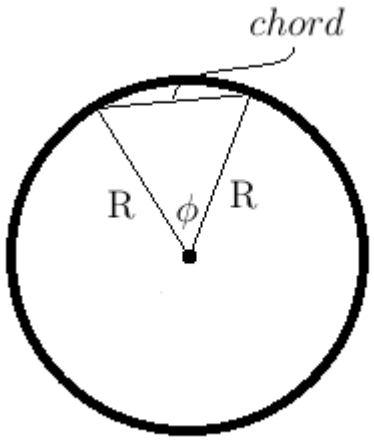

Figure 2.9: Visualization of a chord length on a circle

will herein after be referred to as c_2 and c_3, respectively. The circle traced by the red points in Fig. 2.8 have radii R corresponding to

$$R = r_{hub} + l(n-1) \tag{2.36}$$

where n is the node number, e.g. for Node 2, $n = 2$. Keeping in mind that the s/c hub is rotating about three separate axes, θ is defined as a vector of angles such that

$$\vec{\theta} = \begin{Bmatrix} \psi \\ \theta \\ \phi \end{Bmatrix} \tag{2.37}$$

where

- ψ is the hub rotation about the x-axis
- θ is the hub rotation about the y-axis
- ϕ is the hub rotation about the z-axis

For application in this research, $\vec{\theta}$ denotes the change in angle between the current and previous time step of the algorithm and is defined as

$$\vec{\theta} = \vec{\theta_j} - \vec{\theta_{j-1}} \tag{2.38}$$

Using Eq. (2.36), Eq. (2.38), and Eq. (2.35), the equation used to calculate the chord lengths for Node 2 and Node 3 are such that

$$\vec{c}_{n,k,j} = 2(r_{hub} + l(n-1))sin\left(\tfrac{1}{2}(\vec{\theta_j} - \vec{\theta}_{j-1})\right) \tag{2.39}$$

In this notation, the index k denotes the axes of rotation e.g. $\psi = 1$, $\theta = 2$, and $\phi = 3$. It is important to note that c_1, the chord length corresponding to Node 1, is an unnecessary quantity since Node 1 is always at the origin of the boom's coordinate system.

Because the the coordinate system of the boom is moving but the boom itself is not, the position of the boom in the last time step (the x-z coordinate system in Eq. (2.8)) must be transformed to its corresponding position in the new boom coordinate system (the x'-z' coordinate system in Eq. (2.8)). This can be accomplished using simple geometric and trigonometric identities. Fig. 2.10 examines the isosceles triangle formed

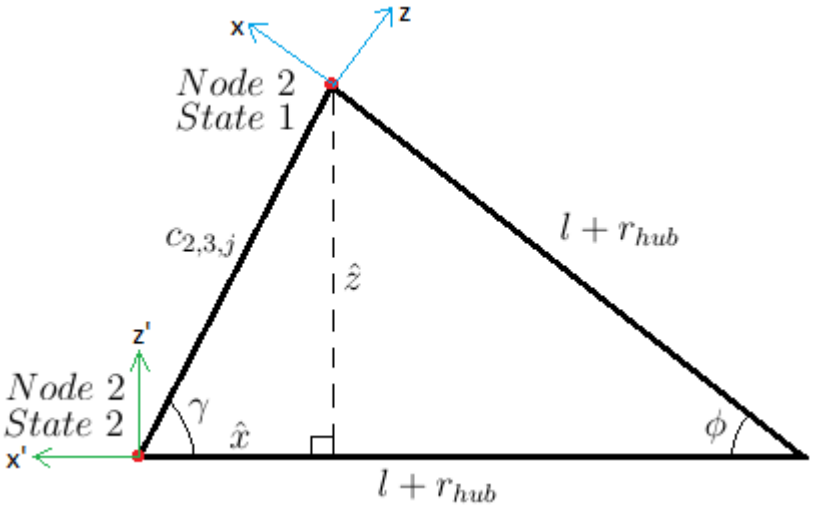

Figure 2.10: The triangle formed by the rotating coordinate system and the chord length

by the origin of the rigid body and Node 2's current and past state, where

- ϕ is the incremental angle of rotation of the rigid body i.e. $\phi_j - \phi_{j-1}$
- $\hat{x}$ is the distance between the x and the x' axes
- $\hat{z}$ is the distance between the z and the z' axes

In order to calculate $\hat{x}$ and $\hat{z}$, the angle γ is required. This can accomplished using the law of sines:

$$\frac{sin\phi}{c_{2,3,j}} = \frac{sin\gamma}{l + r_{hub}} \tag{2.40}$$

and, therefore,

$$\gamma = sin^{-1}\left(\frac{(l + r_{hub})sin\phi}{c_{2,3,j}}\right) \tag{2.41}$$

Once γ is known, the definitions of the trigonometric functions can be used to determine $\hat{x}$ and $\hat{z}$

$$cos\gamma = \frac{\hat{x}_{yaw}}{c_{2,3,j}} \tag{2.42}$$

$$sin\gamma = \frac{\hat{z}_{yaw}}{c_{2,3,j}} \tag{2.43}$$

Referring again to the geometry in Fig. 2.10, it can be seen that the x-z coordinate system is in the negative x' direction and the positive z' direction from the x'-z' coordinate system. This translates to the yaw rotation (ϕ) having a -$\hat{x}$ and a +$\hat{z}$ contribution on the boundary condition of the FEM.

Because the isosceles triangle in Fig. 2.10 used for Node 2 is geometrically similar to the isosceles triangle used for Node 3, $\vec{\Gamma}$ is the same for both Node 2 and Node 3. Generalizing Eq. (2.42) and Eq. (2.43) for both Node 2 and Node 3 yield

$$cos\gamma = \frac{\hat{x}_{n,yaw}}{c_{n,3,j}} \tag{2.44}$$

$$sin\gamma = \frac{\hat{z}_{n,yaw}}{c_{n,3,j}} \tag{2.45}$$

The same method is applied to the other axes of rotation for each of the other booms at Node 2 and Node 3. The angle equivalent to γ for all three axes of rotation is given as

$$\vec{\Gamma}_{k,j} = \begin{Bmatrix} \alpha \\ \beta \\ \gamma \end{Bmatrix}_j = \frac{sin^{-1}((r_{hub} + l)sin(\vec{\theta}_j - \vec{\theta}_{j-1}))}{\vec{c}_{2,k,j}} \tag{2.46}$$

Calculating the boundary condition contributions for the other two axes of rotation is accomplished using the same method as for the yaw rotation. The displacement boundary condition vector P_{bc} is given as

$$P_{bc} = \begin{bmatrix} -\hat{x}_{2,yaw} - \hat{x}_{2,pitch} \\ \hat{y}_{2,pitch} \\ \hat{z}_{2,yaw} \\ 0 \\ 0 \\ -\hat{x}_{3,yaw} - \hat{x}_{3,pitch} \\ \hat{y}_{3,pitch} \\ \hat{z}_{3,yaw} \\ 0 \\ 0 \end{bmatrix} \tag{2.47}$$

where $\hat{x}_{n,pitch}$, $\hat{y}_{n,pitch}$, and β are defined such that

$$cos\beta = \frac{\hat{x}_{n,pitch}}{c_{n,2,j}} \tag{2.48}$$

$$sin\beta = \frac{\hat{y}_{n,pitch}}{c_{n,2,j}} \tag{2.49}$$

$$\beta = sin^{-1}\left(\frac{(l + r_{hub})sin\theta}{c_{2,2,j}}\right) \tag{2.50}$$

The reason that there is no rigid body roll contribution to P_{bc} is because this particular boom is aligned with the x-axis. This means that the motion of the boom is pure rotation, similar to how rigid body pitch would result in a pure rotation of a boom aligned with the y-axis. P_{bc} is added to U_{j-1} to become the initial condition for the next iteration of the FEM.

2.3.3 Zero Elimination of Node 1

Another boundary condition to take into account is that of the cantilevered boom. Since the Node 1 of the boom is always at the origin of the boom's coordinate system, the DOFs corresponding to Node 1 is always zero. This allows us to eliminate all rows and columns of the FEM matrices that correspond to Node 1, greatly simplifying the FEM. The reduced displacement vector is shown:

$$\vec{U}_{[10x1]} = \begin{Bmatrix} u_2 + u_1 \\ v_2 + v_2 \\ w_2 + w_1 \\ \theta_{y2} + \theta_{y1} \\ \theta_{z2} + \theta_{z1} \\ u_2 \\ v_2 \\ w_2 \\ \theta_{y2} \\ \theta_{z2} \end{Bmatrix} = \begin{Bmatrix} u_2 \\ v_2 \\ w_2 \\ \theta_{y2} \\ \theta_{z2} \\ u_3 \\ v_3 \\ w_3 \\ \theta_{y3} \\ \theta_{z3} \end{Bmatrix} \tag{2.51}$$

The reduced mass, stiffness, and damping matrices are also shown:

$$\vec{M}_{[10x10]} = \begin{bmatrix} M_d + M_a & M_b \\ M_c & M_d \end{bmatrix} \tag{2.52}$$

$$\vec{K}_{[10x10]} = \begin{bmatrix} K_d + K_a & K_b \\ K_c & K_d \end{bmatrix} \tag{2.53}$$

$$\vec{B}_{[10x10]} = \begin{bmatrix} B_d + B_a & B_b \\ B_c & B_d \end{bmatrix} \tag{2.54}$$

2.4 Updating the Mass Moment of Inertia

With the displacement vector from the FEM, the mass moment of inertia of the booms can be recalculated to be used in the next iteration of the Euler Moment Equations. First, the mass moment of inertia must be calculated in the boom's local coordinate system. Next, a coordinate transformation must be applied to relate the mass moment of inertia to the rigid body's coordinate system and the Parallel Axis Theorem is applied, as necessary, since the booms are not being rotated about their centroid.

2.4.1 Mass Moment of Inertia Calculation

When the booms are undeformed, the mass moment of inertia can be modeled as that of a thin plate. The equations for this geometry are as follows: Eq. (2.55).

$$\bar{I}_x = \frac{\rho A l}{12}(h^2 + w^2) \tag{2.55a}$$

$$\bar{I}_y = \frac{\rho A l}{12}(l^2 + w^2) \tag{2.55b}$$

$$\bar{I}_z = \frac{\rho A l}{12}(h^2 + l^2) \tag{2.55c}$$

The bar notation indicates that the axis of rotation is about the centroid of the boom. Non-centroidal rotation is discussed in the next section with the Parallel Axis Theorem. Once the booms become a deformed, irregular shape, this method becomes invalid. Instead, the following relations must be used:

$$\bar{I}_{xOy} = \sum_n m_n z_n^2 \tag{2.56a}$$

$$\bar{I}_{yOz} = \sum_n m_n x_n^2 \tag{2.56b}$$

$$\bar{I}_{zOx} = \sum_n m_n y_n^2 \tag{2.56c}$$

In these equations,

- the subscripts xOy, yOz, and zOx denote the reference plane
- m_n is the mass of a given object, to be evaluated as $\rho A l$
- x_n, y_n, and z_n are the perpendicular distances from each reference plane
- n is an index denoting each object to be summed

The mass moment of inertia, defined in Eq. (2.57), can then be calculated about each axis, such that

$$\bar{I}_x = m_n \sum_n (y_n^2 + z_n^2) = \bar{I}_{xOy} + \bar{I}_{zOx} \tag{2.57a}$$

$$\bar{I}_y = m_n \sum_n (z_n^2 + x_n^2) = \bar{I}_{yOz} + \bar{I}_{xOy} \tag{2.57b}$$

$$\bar{I}_z = m_n \sum_n (x_n^2 + y_n^2) = \bar{I}_{zOx} + \bar{I}_{yOz} \tag{2.57c}$$

The values used for x_i, y_i, and z_i are obtained from the displacement vector of the FEM. x_2, y_2, and z_2 are taken from elements 1, 2, and 3 of the displacement vector, respectively, and x_3, y_3, and z_3 are taken from elements 6, 7, and 8 of the displacement vector, respectively. Keeping in mind that x_1, y_1, and z_1 are always zero by construction, Eq. (2.56) can be expanded such that

$$\bar{I}_{xOy} = \rho A l(z_2^2 + z_3^2) \tag{2.58a}$$

$$\bar{I}_{yOz} = \rho A l(x_2^2 + x_3^2) \tag{2.58b}$$

$$\bar{I}_{zOx} = \rho A l(y_2^2 + y_3^2) \tag{2.58c}$$

Thus, Eq. (2.57) can be expanded to

$$\bar{I}_x = \rho Al[(z_2^2 + z_3^2) + (y_2^2 + y_3^2)] \tag{2.59a}$$
$$\bar{I}_y = \rho Al[(x_2^2 + x_3^2) + (z_2^2 + z_3^2)] \tag{2.59b}$$
$$\bar{I}_z = \rho Al[(y_2^2 + y_3^2) + (x_2^2 + x_3^2)] \tag{2.59c}$$

Because the boom lies along the x-axis of its local coordinate system, a factor of l and $2l$ must be added to the x-displacement at Node 2 and Node 3, respectively. This factor is included in Eq. (2.60).

$$\bar{I}_x = \rho Al(z_2^2 + z_3^2 + y_2^2 + y_3^2) \tag{2.60a}$$
$$\bar{I}_y = \rho Al[(x_2 + l)^2 + (x_3 + 2l)^2 + z_2^2 + z_3^2] \tag{2.60b}$$
$$\bar{I}_z = \rho Al[y_2^2 + y_3^2 + (x_2 + l)^2 + (x_3 + 2l)^2] \tag{2.60c}$$

In this research, the shape of the displaced boom (for the sake of the mass moment of inertia calculation) is assumed to be linear in nature. This allows for any point along the boom to be easily interpolated. The perpendicular distance from each axis to the mid-point of each element is used as the average boom element displacement in the mass moment of inertia calculation.

2.4.2 Coordinate Transformation and Parallel Axis Theorem

Referring back to Fig. 2.2, it can be seen that each boom's coordinate system is unique with respect to each other and the rigid body's coordinate system. For example, it cannot be said that mass moment of inertia of the entire system about the x-axis is equal to the sum of rigid body's mass moment of inertia about the x-axis and the booms' mass moments of inertia about the x-axis. To reconcile this, the axes of each coordinate system can be grouped with respect to direction by visual inspection. If the spacecraft were to rotate about its x-axis, the mass moments of inertia actively resisting the rotation would be:

- I_x of the rigid body
- I_x of the red and blue booms
- I_z of the green and orange booms

However, for I_z of the green and orange booms, the Parallel Axis Theorem must be applied since the axes are not collinear. For the red and blue booms, $\bar{I}_x = I_x$. The Parallel Axis Theorem calculates inertia as Eq. (2.61)

$$I = \bar{I} + md^2 \tag{2.61}$$

where d is the perpendicular distance between the two parallel axes. Again, the mass m is evaluated as ρAl. However, since the Parallel Axis Theorem is being applied to two elements, the mass must be multiplied by a factor of two, hence $2\rho Al$. Thus, the corrected mass moment of inertia of the system for a rotation about the x-axis is

$$I_{x,system} = I_{x,rigid} + I_{x,red} + I_{x,blue} + \bar{I}_{z,green} + \bar{I}_{z,orange} + 4\rho Alr_{hub}^2 \tag{2.62}$$

Since the s/c is mirrored for rotations about the x-axis and y-axis, $I_{y,system}$ is easily obtainable from Eq. (2.62):

$$I_{y,system} = I_{y,rigid} + I_{x,green} + I_{x,orange} + \bar{I}_{z,red} + \bar{I}_{z,blue} + 4\rho A l r_{hub}^2 \tag{2.63}$$

$I_{z,system}$ is also obtained through inspection as

$$I_{z,system} = I_{z,rigid} + \bar{I}_{y,red} + \bar{I}_{y,blue} + \bar{I}_{y,green} + \bar{I}_{y,orange} + 8\rho A l r_{hub}^2 \tag{2.64}$$

2.5 Reaction Torque and Drag Force

Because the booms are cantilevered to the rigid body and because they are deflecting, they exert some reaction force on the hub due to material elasticity. This reaction force acts as an external moment on the rigid body. Using the displacement of Node 2 as the tip deflection δ, we establish the relation between the tip deflection and the equivalent force applied at the tip of the boom P [8], given by

$$\delta = \frac{P(2\ell)^3}{3EI_k} \tag{2.65}$$

In this equation, I_k is the area moment of inertia about the bending axis. Because the beam is cantilevered, the force P at the boom tip is also applied on the rigid body. This force creates a moment in the direction opposite the hub rotation. The greater the angular velocity of the hub, the greater the boom deflection and, therefore, the greater the reaction torque in the opposite direction of the hub rotation. Fig. 2.11 shows the equivalent force from the displacement of the tip of a beam.

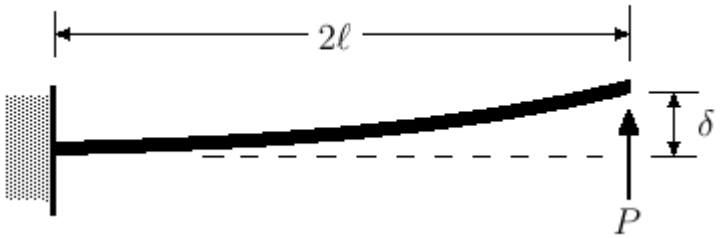

Figure 2.11: Beam deflection at the tip and the equivalent force applied at the tip

Although this does not apply to true MMS conditions, the TableSat is also subject to drag forces and air resistance as it rotates. In order to model this, one starts by first considering the rotation of the hub. If we approximate the hub to be a cylinder of height h_{hub} and radius r_hub rotating in air, we can derive the moment that opposes the hub rotation created from fluid resistance. The shear stress acting on the wall of the cylinder is defined as [9]

$$\tau_{wall} = \mu \frac{du_\theta}{dr} \tag{2.66}$$

where

$$u_\theta = \frac{r_{hub}^2 \omega_z}{r} \tag{2.67}$$

Evaluating Eq. (2.66) for $r = r_{hub}$ yields

$$\tau_{wall}(r = r_{hub}) = -\mu\omega_z \tag{2.68}$$

Integrating the wall shear stress over the entire circumference and height of the cylinder yields

$$F_{wall} = \int^{h_{hub}} \int^{2\pi} \tau_{wall} r_{hub} d\theta dL = -2\pi\mu\omega_z r_{hub} h_{hub} \tag{2.69}$$

The negative sign in this equation denotes that the moment opposes the direction of rotation and can be dropped. Multiplying this wall force by the distance from the distance from the center of rotation r_{hub} yields the moment opposing the hub rotation M_{drag}

$$M_{drag,wall} = 2\pi\mu\omega_z r_{hub}^2 h_{hub} \tag{2.70}$$

The contribution from the top and bottom of the cylinder is defined by

$$M_{drag,top} = \frac{\pi\mu\omega_z r_{hub}^4}{S} \tag{2.71}$$

where S is the thickness of fluid above/below the top/bottom of the cylinder. Because the TableSat is in a large open space, S is very large and thus the drag contribution from the top/bottom of the cylinder is negligible.
The drag contribution from the booms follows the relation for the drag force F_D on a thin plate, given by

$$F_{D,boom} = \frac{C_D A \rho v^2}{2} \tag{2.72}$$

where

- C_D is the drag coefficient for a thin plate $C_D = 1.9$
- A is the area of the thin plate normal to the flow
- ρ is the density of the fluid (air)
- v is the free stream velocity of the fluid

The free stream velocity is a function of the rotational velocity of the hub ω_z, as shown:

$$v = r\omega_z \tag{2.73}$$

The moment opposing rotation is then the integral of the drag force $F_{D,boom}$ over the length of the boom as a function of the distance from the center of rotation, given by

$$M_{drag,boom} = \int_{r_{hub}}^{r_{hub}+2\ell} F_{D,boom} dr = \int_{r_{hub}}^{r_{hub}+2\ell} \frac{C_D A \rho \omega_z^2 r^2}{2} dr \tag{2.74}$$

Evaluating this integral yields

$$M_{drag,boom} = \frac{C_D A \rho \omega_z^2}{6}[(r_{hub} + 2\ell)^3 - r_{hub}^3] \tag{2.75}$$

Keeping in mind that there are four booms and that the top and bottom of the TableSat have a negligible contribution to the drag, the total moment opposing rotation due to drag is given by

$$M_{drag} = M_{drag,wall} + 4M_{drag,boom} \tag{2.76}$$

A correction factor to account for the fluid friction of the air bearing needs to be empirically defined. This will be discussed in Chapter 5.

Chapter 3

Controller Design

In this chapter, the design of three different controllers is discussed in detail. The motivation for designing these controllers stems from being able to reject any disturbances that may potentially jeopardize the success of the mission, such as disrupting the formation of the s/c or moving the s/c out of a desired orbit. The three control methods are chosen from different areas of controls: classical controls, nonlinear controls, and optimal control.

3.1 Proportional Integral Derivative (PID) Control

This classical feedback control method continuously attempts to minimize the error of the system by weighting different terms using Eq. (3.1) [10].

$$\vec{M}_o = \vec{K}_P\vec{e}(t) + \vec{K}_I\int_t \vec{e}(t) + \vec{K}_D\frac{d}{dt}\vec{e}(t) \tag{3.1}$$

where

- $\vec{M}_o$ is the output of the controller and the external moment applied on the rigid body via thrusters
- $\vec{e}(t)$ is the error function
- $\vec{K}_P$ is the matrix of proportional gains
- $\vec{K}_I$ is the matrix of integral gains
- $\vec{K}_D$ is the matrix of derivative gains

The error is defined as the difference between the desired value of a state and the actual measured value of a state. In this research, the state of interest is the angular velocity about all three axes, as defined in Eq. (3.2).

$$\vec{e}(t) = \begin{Bmatrix} e_x \\ e_y \\ e_z \end{Bmatrix} = \begin{Bmatrix} \omega_{x,desired} - \omega_{x,measured} \\ \omega_{y,desired} - \omega_{y,measured} \\ \omega_{z,desired} - \omega_{z,measured} \end{Bmatrix} \tag{3.2}$$

PID control is a linear control method and therefore will not behave properly with a nonlinear system. Because Euler's Moment Equations are a nonlinear system of equations, they are linearized for implementation of the PID controller. The plant (Euler's Moment Equations) is linearized using a Taylor series expansion, such that

$$f(x) \approx f(\bar{x}) + \frac{d}{dx} f(x)\Big|_{x=\bar{x}} (x - \bar{x}) + H.O.T. \tag{3.3}$$

In this equation,

- $\bar{x}$ is the user-defined point about which the linearized model operates
- $H.O.T.$ are higher order terms that are approximated as being negligible

Expanding this for the multi-variable plant about the point $\bar{\omega} = (\bar{\omega}_x, \bar{\omega}_y, \bar{\omega}_z)$ yields

$$\begin{aligned} f(\omega_x, \omega_y, \omega_z) \approx f(\bar{\omega}_x, \bar{\omega}_y, \bar{\omega}_z) &+ \frac{\partial}{\partial \omega_x} f(...)\Big|_{\omega=\bar{\omega}} (\omega_x - \bar{\omega}_x) \\ &+ \frac{\partial}{\partial \omega_y} f(...)\Big|_{\omega=\bar{\omega}} (\omega_y - \bar{\omega}_y) + \frac{\partial}{\partial \omega_z} f(...)\Big|_{\omega=\bar{\omega}} (\omega_z - \bar{\omega}_z) \end{aligned} \tag{3.4}$$

Evaluating this expansion for Eq. (2.1) yields the linearized plant

$$\dot{\omega}_x = \frac{1}{I_x}[M_x - \bar{\omega}_z(I_z - I_y)(\omega_y - \bar{\omega}_y)] \tag{3.5a}$$

$$\dot{\omega}_y = \frac{1}{I_y}[M_y - \bar{\omega}_z(I_x - I_z)(\omega_x - \bar{\omega}_x)] \tag{3.5b}$$

$$\dot{\omega}_z = \frac{M_z}{I_z} \tag{3.5c}$$

For this application, the linearization point $\bar{\omega}$ is defined as the desired point ω_{des}, since this will be the point where the system will be expected to operate for most of the time. Because this is a spin-stabilized s/c, the desired rates of roll and pitch rotation will always be zero and the rate of yaw rotation will be some constant rate $\bar{\omega}_z$, making the linearization operating point

$$\bar{\omega} = \begin{Bmatrix} 0 \\ 0 \\ \bar{\omega}_z \end{Bmatrix} \tag{3.6}$$

and, therefore, the linearized plant

$$\dot{\omega}_x = \frac{1}{I_x}[M_x - \bar{\omega}_z(I_z - I_y)\omega_y] \tag{3.7a}$$

$$\dot{\omega}_y = \frac{1}{I_y}[M_y - \bar{\omega}_z(I_x - I_z)\omega_x] \tag{3.7b}$$

$$\dot{\omega}_z = \frac{M_z}{I_z} \tag{3.7c}$$

In an effort to simulate the exact TableSat conditions for experimental validation purposes, the gains of the PID controller were chosen to be the same as that of the TableSat (which were chosen by trial and error). Only proportional (P) controllers are needed for the x-axis and y-axis angular accelerations. For the z-axis acceleration,

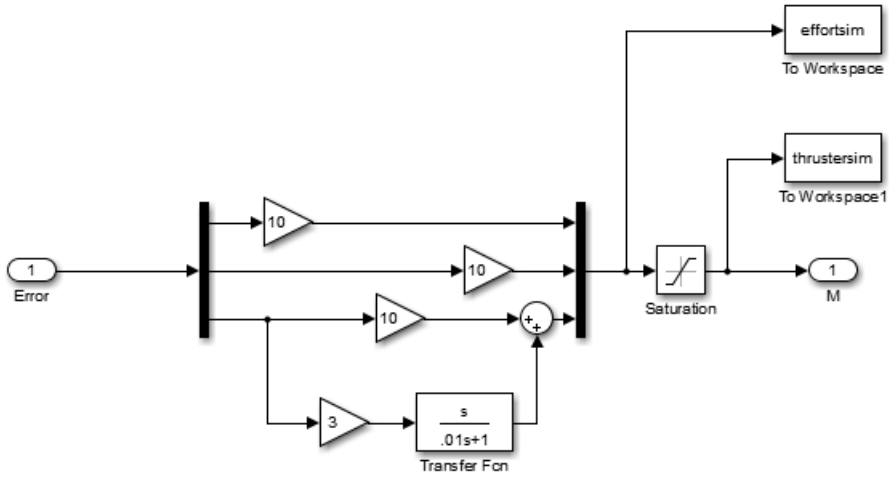

Figure 3.1: Simulation model of the P and PD controllers

a proportional-derivative (PD) controller is used. The transfer function of the PD controller $G_{PD}(s)$, relating the moment input to the error signal, is given by

$$G_{PD}(s) = K_P + K_D s \tag{3.8}$$

However, the simulation software does not accept this as a proper transfer function. Therefore, the transfer function must be linearized:

$$G_{PD}(s) = K_P + K_D \frac{s}{cs + 1} \tag{3.9}$$

where c is a user-defined coefficient that is used in the approximation. This coefficient is chosen as $c = 0.01$. The simulation model for the three P/PD controllers is shown in Fig. 3.1.

3.2 Sliding Mode Control (SMC)

Sliding mode control (SMC) is a nonlinear control method that tracks a desired state despite modeling imprecisions such as parametric uncertainty or unmodeled dynamics. Examples of each of these pertaining to this research include the time-varying mass moment of inertia and simplified model of friction/drag, respectively. SMC works by dividing the phase plane into separate control structures and forcing the trajectory of the states towards a sliding surface $s(t)$ between structures, as shown in Fig. 3.2. This is known as the reaching phase. Once the state reaches the surface $s(t)$, it slides towards the desired point while remaining on the sliding surface. This is known as the sliding phase. A depiction of the state's trajectory is shown in Fig. 3.3. The drawback of SMC, however, is that the high performance comes at the price of high control activity (frequency) and, therefore, control effort. Not only is there significant control effort, but this significant control activity can actually excite the unmodeled dynamics of the system, creating an effect called chattering, as shown in Fig. 3.4. One solution to inhibit the effects of chattering is to create a numerical boundary layer along the sliding surface. This boundary layer effectively increases the thickness of the sliding surface such that the state is not trying to track an infinitesimally thin surface.

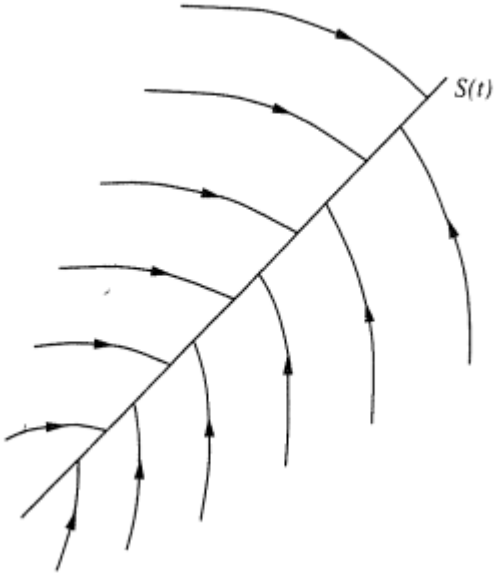

Figure 3.2: The sliding condition of the states from any given initial condition to the sliding surface

Since the trajectory of the state is driven to zero along the sliding surface, it appropriate to define the sliding surface as a function of the error, since it is also desirable for the error to be driven to zero. In this research, the sliding surface is defined as the error itself.

$$\vec{s}(e) = \vec{e}(t) \tag{3.10}$$

Then the equation for the first order sliding mode controller is

$$\vec{u}(t) = \mathcal{K} sat\left(\frac{\vec{s}(e)}{\phi}\right) + \dot{\vec{s}}(e) \tag{3.11}$$

In this equation:

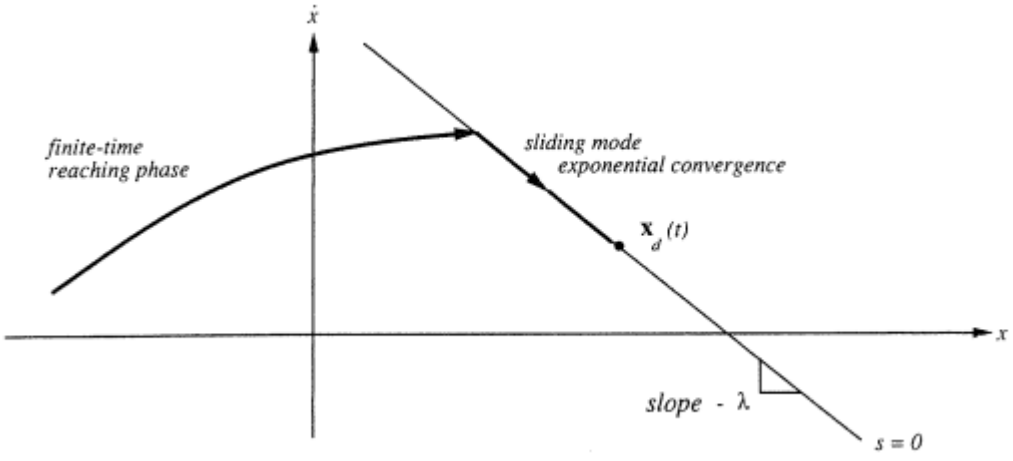

Figure 3.3: Depiction of reaching and sliding phases [11]

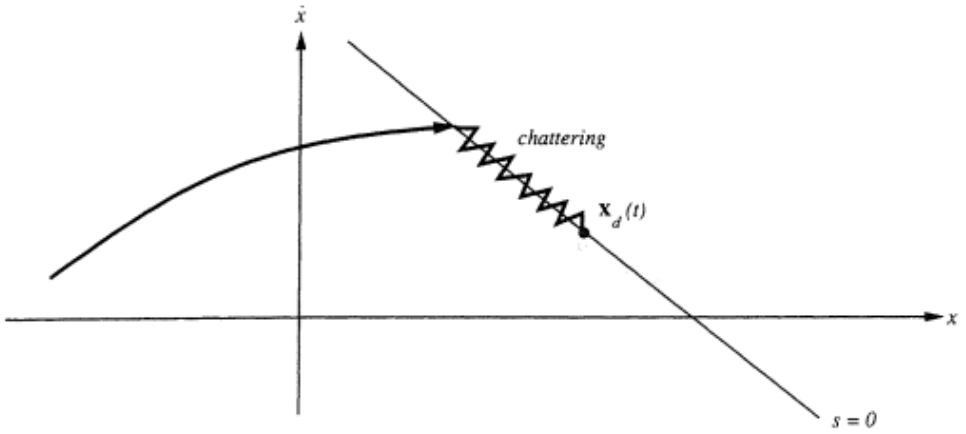

Figure 3.4: The adverse effects of high control activity on unmodeled dynamics [11]

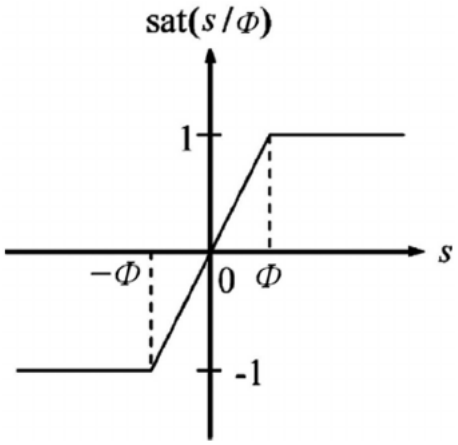

Figure 3.5: The discontinuous saturation function

- $\mathcal{K}$ is a user-defined gain
- $sat()$ is the discontinuous saturation function shown in Fig. 3.5
- *phi* is the user-defined boundary layer thickness

The Simulation model for the SMC is shown in Fig. 3.6.

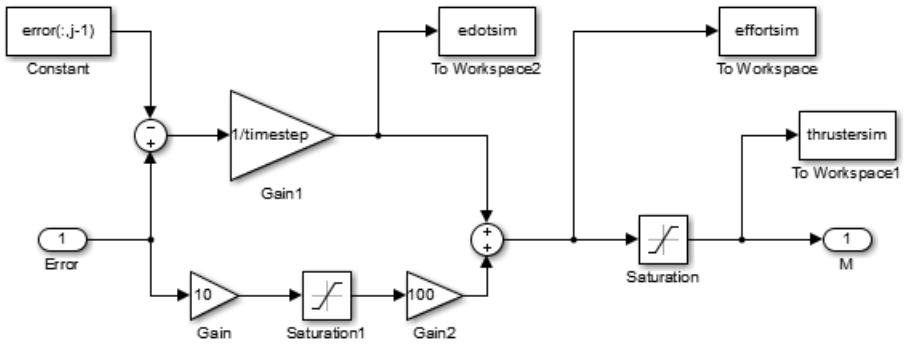

Figure 3.6: Simulation model of the Sliding Mode Controller

3.3 Linear Quadratic Regulator (LQR)

The LQR controller is an optimal control method that optimizes the state space model of a system according to a cost function. This cost function is user-defined and dictates which cost metrics are of greater priority. For example, if the objective must be achieved in minimum time, the cost function can be defined as such at the cost of high fuel expenditure. Conversely, if minimum fuel consumption is the highest priority, the cost function can be defined as such at the cost of time. These weights on fuel and time can be manipulated to find an optimal ratio for a specific application. The cost function J for the LQR problem is defined such that

$$J = \frac{1}{2}\int_0^\infty \vec{x}^T(t)\vec{Q}\vec{x}(t) + \vec{u}^T(t)\vec{R}\vec{u}(t)dt \tag{3.12}$$

where

- $\vec{x}(t)$ is the state vector
- $\vec{u}(t)$ is the control input vector
- $\vec{Q}$ is the user-defined state cost matrix (weight)
- $\vec{R}$ is the user-defined input cost matrix (weight)

If the linear system is defined by the state space equation

$$\dot{\vec{x}}(t) = \vec{A}\vec{x}(t) + \vec{B}\vec{u}(t) \tag{3.13}$$

then the control input is defined by

$$\vec{u}(t) = -\vec{\mathcal{K}}(t)\vec{x}(t) \tag{3.14}$$

where

$$\vec{\mathcal{K}} = \vec{R}^{-1}\vec{B}^T\vec{S} \tag{3.15}$$

Here, $\vec{S}$ is found from solving the algebraic Riccati equation

$$0 = \vec{S}\vec{A} - \vec{A}^T\vec{S} + \vec{S}\vec{B}\vec{R}^{-1}\vec{B}^T\vec{S} - \vec{Q} \tag{3.16}$$

The LQR controller assumes the system to be in the state space form. The state vector is defined as

$$\vec{x}(t) = \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \tag{3.17}$$

and the input vector as

$$\vec{u}(t) = \begin{Bmatrix} M_x \\ M_y \\ M_z \end{Bmatrix} \tag{3.18}$$

The state space model then becomes

$$\begin{Bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{Bmatrix} = \begin{bmatrix} 0 & -\bar{\omega_z}\dfrac{I_z - I_y}{I_x} & 0 \\ -\bar{\omega_z}\dfrac{I_x - I_z}{I_y} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} + \begin{bmatrix} \dfrac{1}{I_x} & 0 & 0 \\ 0 & \dfrac{1}{I_y} & 0 \\ 0 & 0 & \dfrac{1}{I_z} \end{bmatrix} \begin{Bmatrix} M_x \\ M_y \\ M_z \end{Bmatrix} \tag{3.19}$$

Fig. 3.7 shows the simulation model of the LQR controller.

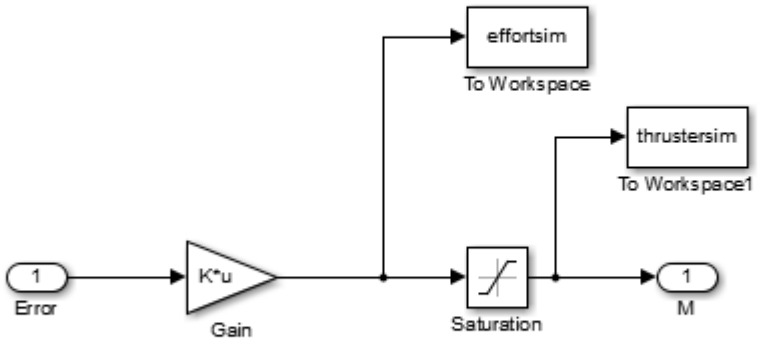

Figure 3.7: Simulation model of the LQR controller

Chapter 4

Experimental Platform and Verification

The experimental platform used to validate this research is NASA's MMS TableSat Generation IC (herein known as TableSat), shown in Fig. 4.1. In this chapter, the

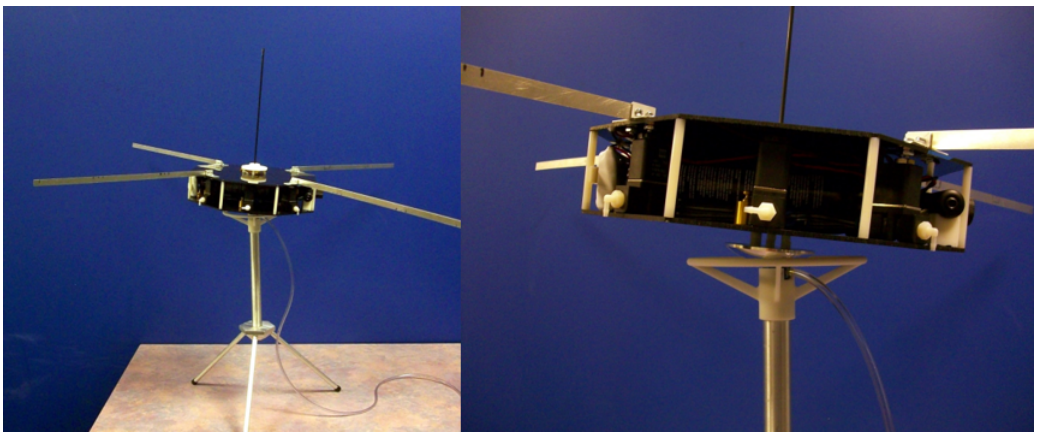

Figure 4.1: TableSat IC

design of TableSat is discussed in detail, as well as the control system that drives it. Measurement methods and sensors are also discussed. Lastly, results from four different tests run on the TableSat are presented. These tests incorporate: constant external moment on the rigid body, impulse force on a boom, spinning up from rest, and steady state disturbance.

4.1 TableSatIC Design

TableSat is a limited three degree-of-freedom, small-scale model of NASA's MMS s/c. It was constructed by and explained in full detail in Chabot et. al [12]. It features full rotation about its z-axis and limited nutation (rotation about its x-axis and y-axis). In order to facilitate nearly frictionless motion, the TableSat is mounted on a parabolic mirror that sits on an air bearing. The air bearing is mounted on a tripod stand and is supplied air from a small compressed air tank. The 3D-printed bumper inhibits full nutation but protects the TableSat from over-rotating and displacing itself from the air bearing.

TableSat has four booms mounted to its frame. These booms are thin flat bars constructed from steel that prevent sagging due to gravity but allow for deflection within the spin plane of the model. Measurement of this deflection is a key component of this research. The entire model is scaled down from the actual MMS based on the natural frequency of the booms.

TableSat's rotation is driven by six pneumatic actuators: two opposing thrusters each for rotation about the x-axis and y-axis and two cooperating thrusters for rotation about the z-axis. This results in the TableSat being able to counter rotation in either the positive or negative direction for rotation about the x-axis and y-axis, but not the z-axis. If the model overshoots its desired rate of yaw rotation, its only means of slowing down are friction and air resistance. The air supply for the thrusters comes from an onboard compressed air tank. The controller for the pneumatic thrusters is on board an Arduino Mega 2560 [13].

4.2 TableSat Controllers

As mentioned in the previous chapter, the controller specifications of the modified hybrid algorithm are chosen to match those of the TableSat. In this section, the specifications of the TableSat controllers are discussed. The code for the TableSat is found in Chabot et al. [12].

4.2.1 Experimental PID Controller

As mentioned previously, two P controllers and one PD controller are used to control the TableSat's rotation about the x and y-axes and the z-axis, respectively. The two P controllers have a gain of $K_P = 10$. The PD control gains are $K_P = 10$ and $K_D = 3$, making $G_{PD}(s)$, the transfer function of the PD controller,

$$G_{PD}(s) = 10 + 3\frac{s}{0.01s + 1} \tag{4.1}$$

4.2.2 Experimental LQR Controller

The LQR controller onboard the TableSat uses a slightly different model than the modified hybrid algorithm, but achieves the same goal. The TableSat LQR model features five states: the three body rates and two quaternion states for attitude control. The hybrid algorithm does not focus on attitude control, only rotation rate control, and therefore excludes the two quaternion states. However, this does not affect the rate control of the two different controllers. The state space model for the TableSat LQR controller is given by

$$\begin{Bmatrix} \dot{i} \\ \dot{j} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{Bmatrix} = \begin{bmatrix} 0 & \frac{\omega_{z,des}}{2} & 0 & 0 & 0 \\ \frac{-\omega_{z,des}}{2} & 0 & 0 & 0 & \frac{-scalar_{des}}{2} \\ 0 & 0 & xx & xx & xx \\ 0 & 0 & xx & xx & xx \\ 0 & 0 & xx & xx & xx \end{bmatrix} \begin{Bmatrix} i \\ j \\ \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{I_x} & 0 & 0 \\ 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix} \begin{Bmatrix} M_x \\ M_y \\ M_z \end{Bmatrix} \tag{4.2}$$

where the elements within the box are such that

$$\omega_{z,des} I^{-1} \begin{bmatrix} I_{yx} & I_{yy} - I_{zz} & 2I_{yz} \\ I_{zz} - I_{xx} & I_{xy} & -2I_{xz} \\ -I_{yz} & I_{xz} & 0 \end{bmatrix} \tag{4.3}$$

Eq. (4.3) is obtained from the linearized plant when the full mass moment of inertia tensor (including off diagonal terms) is used where I is the moment of inertia tensor given by

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{4.4}$$

The Q and R matrices are chosen such that $Q \in I^5$ and $R \in I^3$. With this information, the LQR gain matrix $mathcal K$ is found to be

$$\mathcal{K} = \begin{bmatrix} 0.0734 & 0.0833 & 1.0038 & -0.0367 & -0.0183 \\ 0.1122 & -0.0548 & -0.0367 & 0.9935 & -0.0338 \\ 1.3700 & -0.3087 & -0.0183 & -0.0338 & 1.1498 \end{bmatrix} \tag{4.5}$$

4.3 Sensors and Measurement

Aboard the TableSat there are two key sensors: the Razor Inertial Measurement Unit (IMU) and the sensors for measuring the boom deflections. For the latter, three different methods are explored: accelerometers, capacitive displacement sensors, and strain gauges. Because the TableSat is constantly rotating, it cannot be tethered to a computer via USB cable for serial transmission of data. Therefore, XBees [14] must be used to wirelessly transmit all data to the computer for post-processing.

4.3.1 Razor IMU

The Razor IMU is a nine degree-of-freedom measurement unit featuring a three-axis gyroscope, a three-axis accelerometer, and a three-axis magnetometer [15]. The IMU is responsible for keeping track of the attitude and rate of rotation of the TableSat. This information is fed back to the TableSat's Arduino for processing. The IMU is mounted at the center of the top pane of the TableSat with the z-axis orthogonal to the pane and the x and y-axes aligned with the booms.

4.3.2 Accelerometers

This method of boom deflection measurement was the method used in the original hybrid algorithm by Medas and Thein. Each boom is fitted with three accelerometers: one at the base of the boom, one at the midpoint of the boom, and one at the tip of the boom, as shown in Fig. 4.2. As the boom deflects, the accelerometers measure the acceleration experiences at all three of the points on the boom. The code and wiring diagram for the accelerometers is shown in Appendix B. The output from these accelerometers is integrated twice with respect to time to obtain the displacement at any given time during the experiment. A few problems arise from this procedure, however. The lateral accelerations experienced by the booms are very small. Coupled with the sensor noise, the lateral accelerations are essentially undetectable. Furthermore,

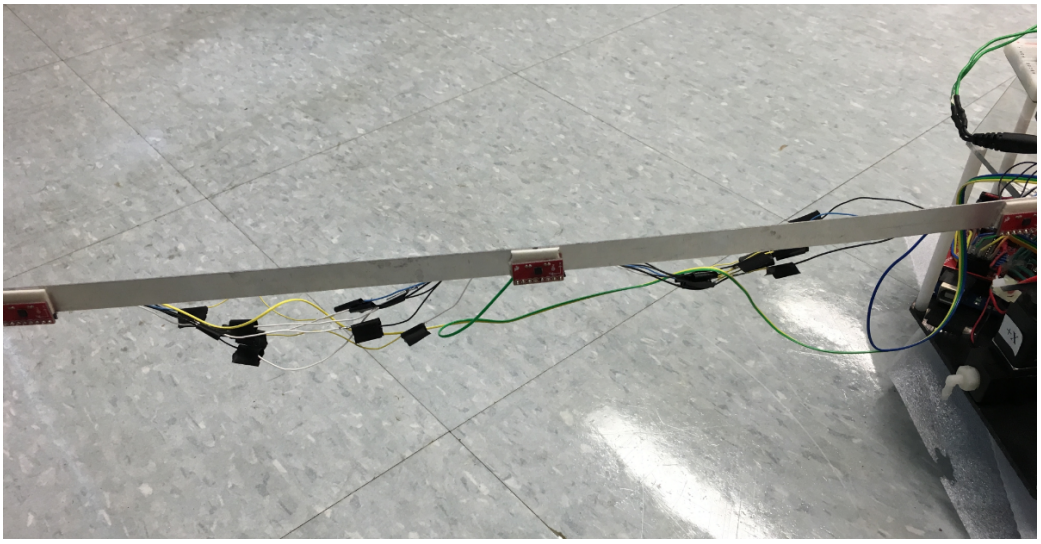

Figure 4.2: The experimental set up with accelerometers mounted on the booms

since the TableSat is constantly rotating, the accelerometers experience a constant centripetal force which is much larger than the lateral acceleration, skewing the signal even more. When the signal is integrated twice, the noise and centripetal force cause the signal to have excessively large magnitudes. Even after subtracting the centripetal force from the accelerometer output, the signal was still extremely large due to the noise, shown in Fig. 4.3. Another adverse affect of the accelerometers is their weight. The weight of the accelerometers themselves and all their wires surely have an effect on the natural frequency of the booms in regard to the FEM, the mass moment of inertia in regards to the rigid body dynamics, and the drag force.

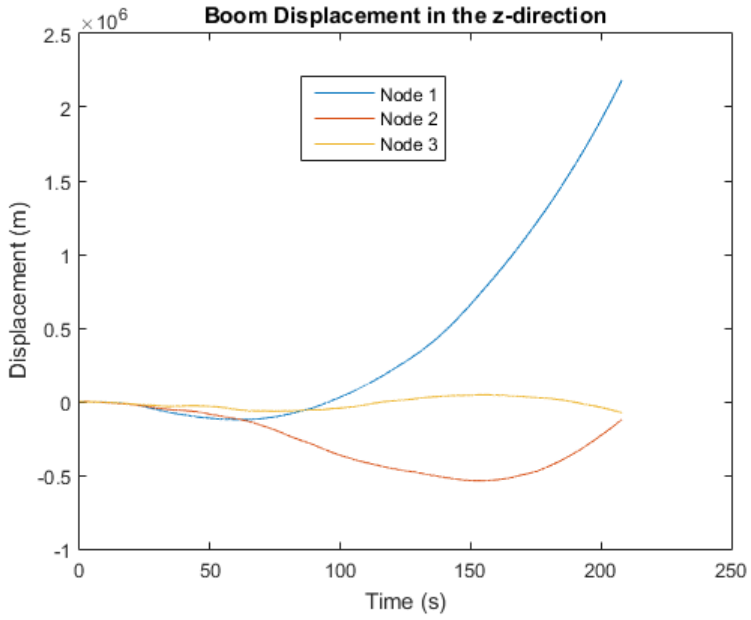

Figure 4.3: Data from the accelerometers during the spin up from rest of the TableSat

4.3.3 Capacitive Displacement Sensor

This method is explored in response to the fact that the accelerometers and their wiring were affecting the TableSat system parameters. The capacitive displacement sensor is a non-contact method of measuring the deflection of the booms. In theory, this sensor measures the capacitance between two conductive plates (the boom and a piece of metal foil) according to

$$C = \frac{k\epsilon_0 A}{d} \tag{4.6}$$

where

- k is the relative permittivity of the dielectric between the conductive plates
- ϵ_0 is the permittivity of free space
- A is the area of the plates
- d is the distance between the plates

The metal foil is mounted to a piece of acrylic that is mounted to the rigid body and runs parallel to the boom, as shown in Fig. 4.4. As the boom deflects, the distance between the boom and the foil d varies and thus varies the capacitance between the two conductors. This sensor utilizes the CapSense library for the Arduino, which turns two of the Arduino's pins into capacitive sensors [16]. The Arduino code and wiring diagram for this sensor are shown in Appendix B. The output of the sensor is given in bits. However the documentation for this library provides no real information on what this value means physically. In order to calibrate the sensor, the variable capacitor is replaced with a series of known capacitors, and the bit value recorded. There were problems experienced with this sensor as well, however. At best, the sensor was unreliable and the data obtained was inconsistent. This is most likely due to interference from outside electric and magnetic fields. Even when the boom experienced large deflections, the sensor is not sensitive enough to detect these changes. The sensor data was also very noisy, resulting in essentially useless data, shown in Fig. 4.5. Although the sensor was able to measure the deflection of the boom at its greatest point, it was not able to capture the other oscillations of the boom's motion, making it impractical for this application.

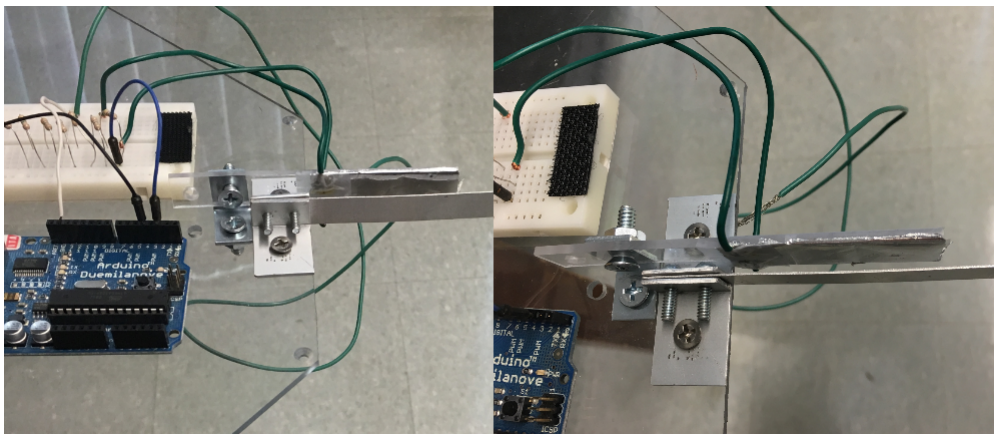

Figure 4.4: Experimental set up with the capacitive displacement sensor mounted

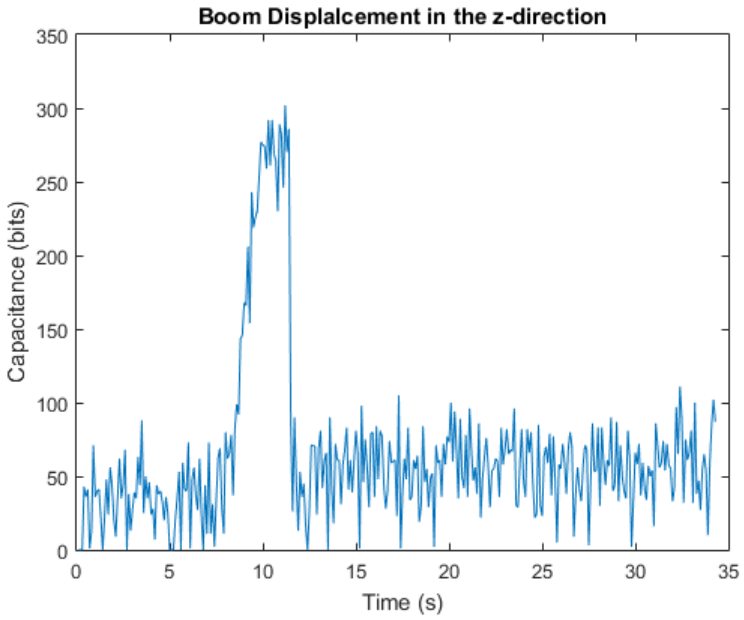

Figure 4.5: Data from the capacitive sensor for a large boom deflection

4.3.4 Strain Gauges

This method utilizes OMEGA SGD-2/350-LY11 strain gauges to measure the strain created from the bending of the booms [17]. In order to achieve the highest possible sensitivity, four strain gauges are used. A full Wheatstone Bridge configuration is used with two strain gauges on each side of the boom. In this configuration, two strain gauges are always equally in compression while the other two are equally in tension, depending on the direction of the deflection. The full Wheatstone bridge is shown in Fig. 4.6. Since each strain gauge has a variable resistance, when the boom deflects, the Wheatstone Bridge becomes unbalanced and a voltage difference can be measured across the bridge. This voltage difference is very small and so an amplifier is used to make the signal detectable by the Arduino. The amplifier used is an instrumentation amplifier from Texas Instruments, the INA125P [18]. The INA125P is a 16-pin, high-gain, low-noise, precision amplifier specifically used for strain gauge and other instrumentation applications. The output from the Wheatstone Bridge is amplified through the INA125P before being processed by the Arduino [19]. The code and wiring diagram for the strain gauge circuit is shown in Appendix B and the configuration is shown in Fig. 4.7. When the free response of the boom is measured after an initial displacement, the results from the strain gauge are significantly improved over the previous two sensing methods. The plot from the free response is shown in Fig. 4.8. The sensitivity of the sensor to deflection and the magnitude of noise are acceptable for making reasonably accurate measurements of strain in the beam due to deflection.

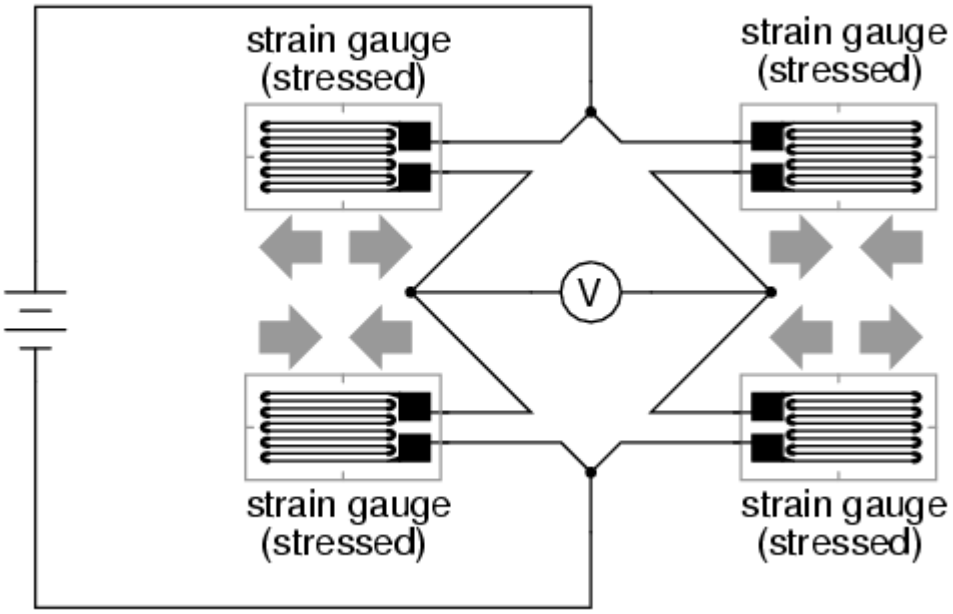

Figure 4.6: Four strain gauges in a full Wheatstone Bridge configuration

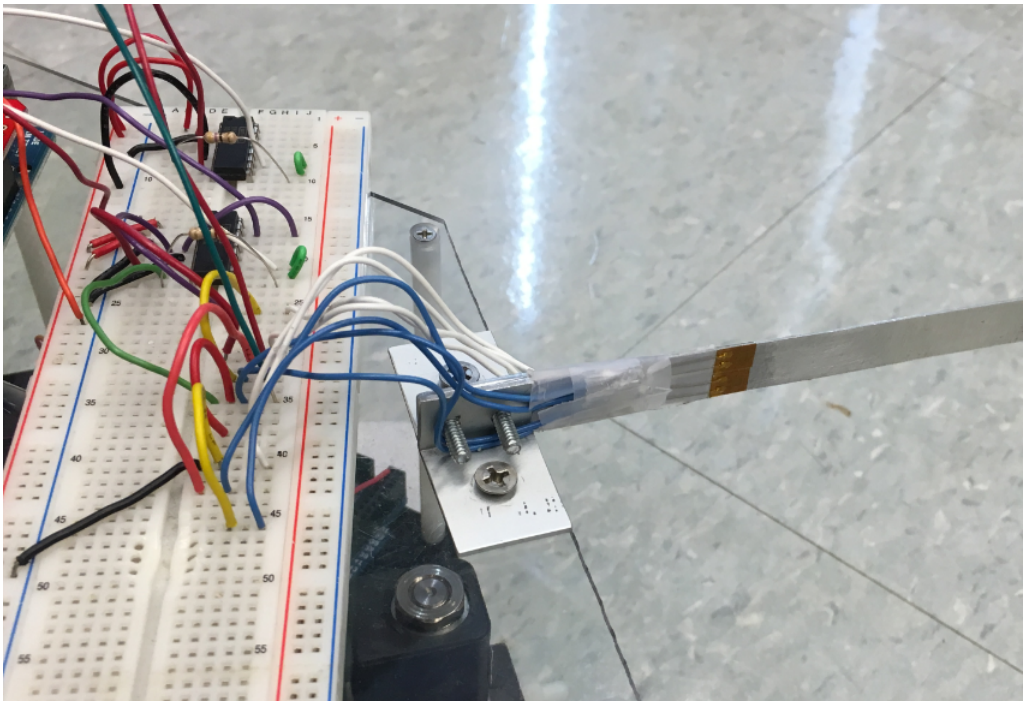

Figure 4.7: Strain gauge-amplifier circuit mounted on the TableSat

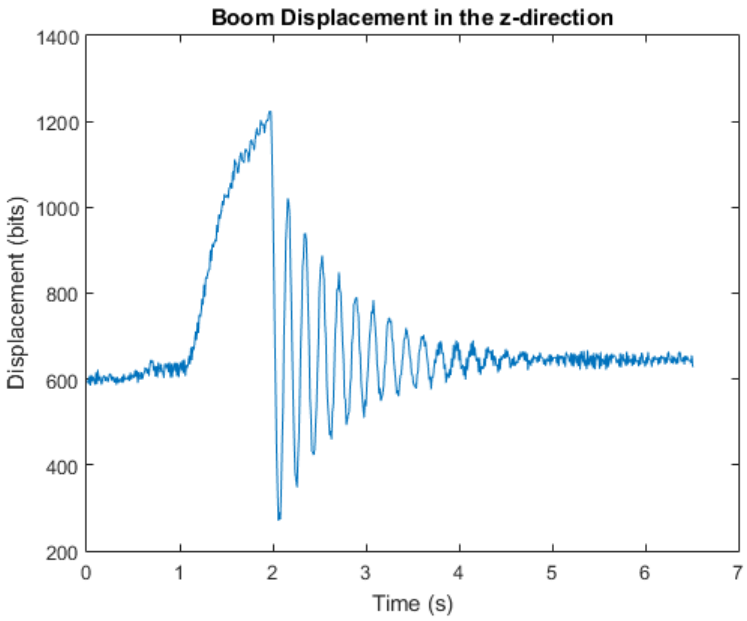

Figure 4.8: Plot of strain in boom during free response after an initial displacement

4.4 Strain Gauge Calibration

In order to calibrate the output from the strain gauge sensor into a physical quantity, the TableSat yaw thrusters are switched into the ON state for the duration of the experiment. The high steady state angular velocity of the rigid body results in a constant, measurable boom deflection at steady state. With the deflection of the boom recorded in bits, the deflection is reproduced under static conditions by reproducing the deflection in bits. The deflection is then measured in meters while in this state. With this information, the sensitivity of the strain gauge sensor is found to be $0.000465\,^{\text{meters}}/_{\text{bit}}$. Since the zero deflection point of the boom is not at zero bits, the actual zero point is multiplied by the sensitivity to find the zero point correction factor. The entire output in bits is then multiplied by the sensitivity to obtain the output of the sensor in meters of deflection. The zero point correction factor is then subtracted from the output. This procedure is repeated for every experiment with the TableSat and the results are presented in the following section.

4.5 Results

In this section, the results from experimentation on the TableSat are presented. Discussion of the presented results and comparison with the hybrid modeling algorithm are presented in Chapter 6.

4.5.1 Constant External Moment

In this experiment, the yaw thrusters of the TableSat are switched into the ON state for the duration of the experiment. No controllers are used. The TableSat eventually reaches a steady state velocity large enough to create a measurable deflection in the booms. Fig. 4.9 and Fig. 4.10 show the rigid body rotation and boom displacement, respectively. The rigid body's angular velocity is shown in Fig. 4.9. The strain gauge data, shown in Fig. 4.10, is relatively noisy due to the small deflections of the booms. However, there is a clear trend in the data showing that as the angular velocity of the rigid body increases, the deflection in the boom increases as well. As the rigid body begins to reach and settle at its steady state velocity (about 4.1 rad/s), the deflection in the boom also reaches a steady state, constant displacement of about 0.02 meters (the average of the noisy signal).

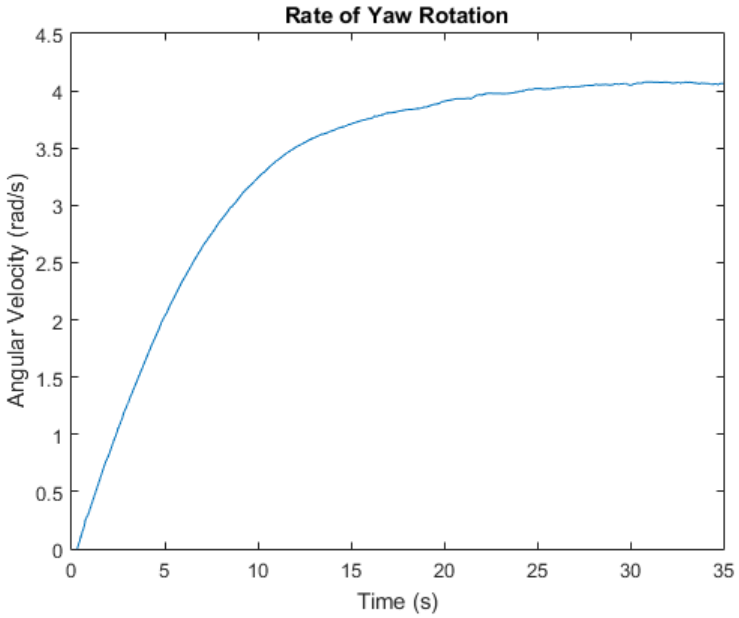

Figure 4.9: TableSat yaw rotation rate under constant external moment

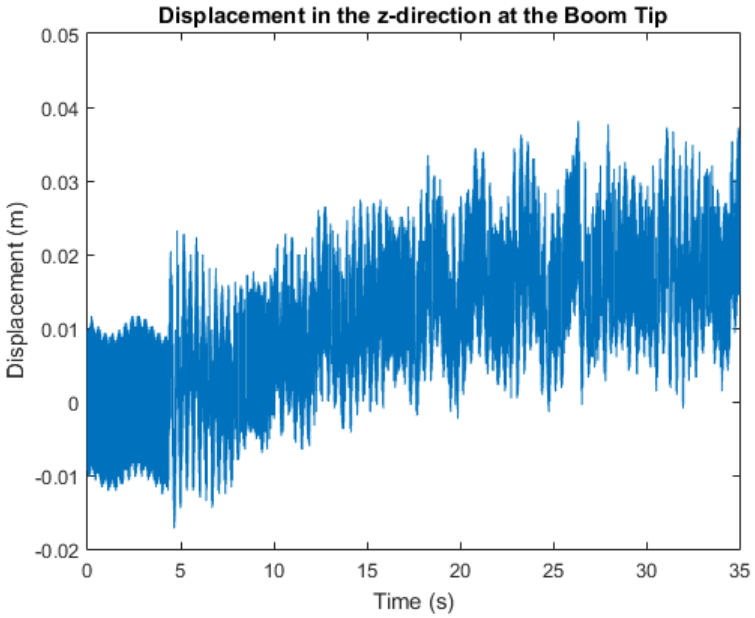

Figure 4.10: Displacement of the boom tip in the z-direction under constant external moment

4.5.2 Impulse Force and Free Response

In this experiment, the controller is turned off and an impulse force is applied to the tip of a boom. The resulting free responses of the rigid body and booms are recorded and shown in Fig. 4.11 and Fig. 4.12, respectively. Looking first as the boom deflection

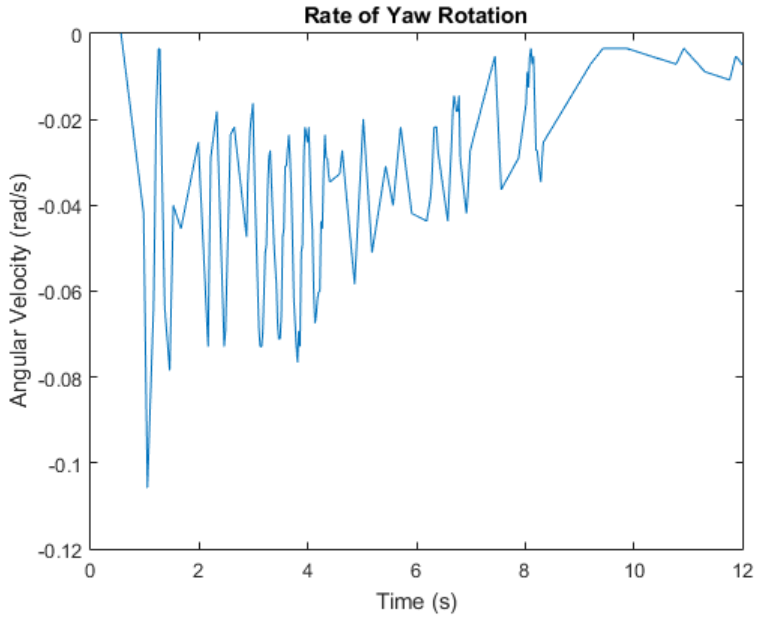

Figure 4.11: TableSat yaw rotation rate after impulse force on boom

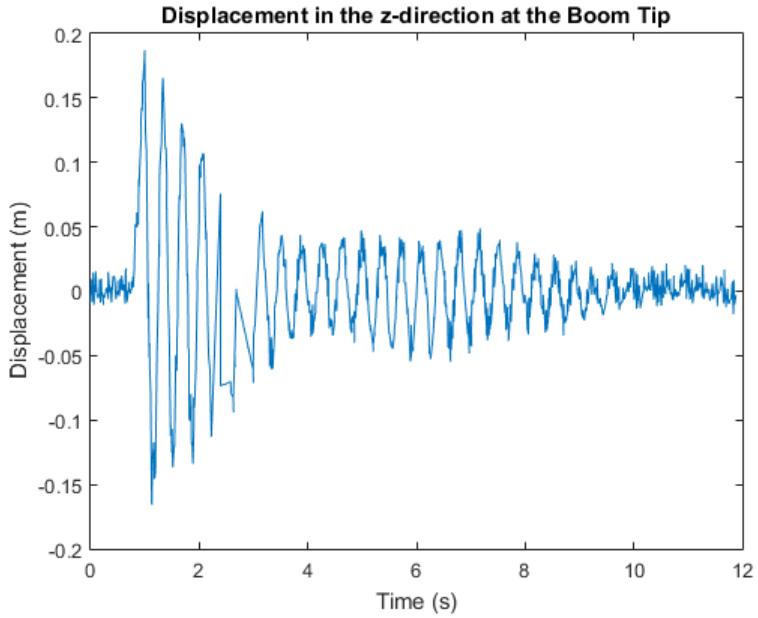

Figure 4.12: Displacement of the boom tip in the z-direction after impulse force on boom

data in Fig. 4.12, it can be seen that when an impulse force is placed on the tip, the strain gauge capture the oscillations of the boom's free response well. The missing data points at $t \approx 3s$ are lost due to a loss of the wireless transmission of data at

that time. However, the trend in the data is clear in that the oscillations in the boom that the force acted upon excites vibrations in the other booms. At $t \approx 4s$, all the booms are in resonance with each other as a result of transferring vibrations to each other. This can be seen where the oscillations reach a local minimum and then start to increase again before decaying to zero. The boom motion also translates motion to the rigid body, shown in Fig. 4.11. In this plot, it can be seen that at the time the force is applied to the boom, the rate of rotation of the rigid body spikes and then proceeds to decay back to zero. This signal has some noise in it, but the noise is due to the very small rate of rotation.

4.5.3 Spin Up from Rest

In this experiment, the TableSat begins at rest and then increases its spin rate up to the desired value of 10 revolutions per minute (1.047 radians per second) and is driven by the LQR controller. Fig. 4.13 and Fig. 4.14 show the results from this experiment.

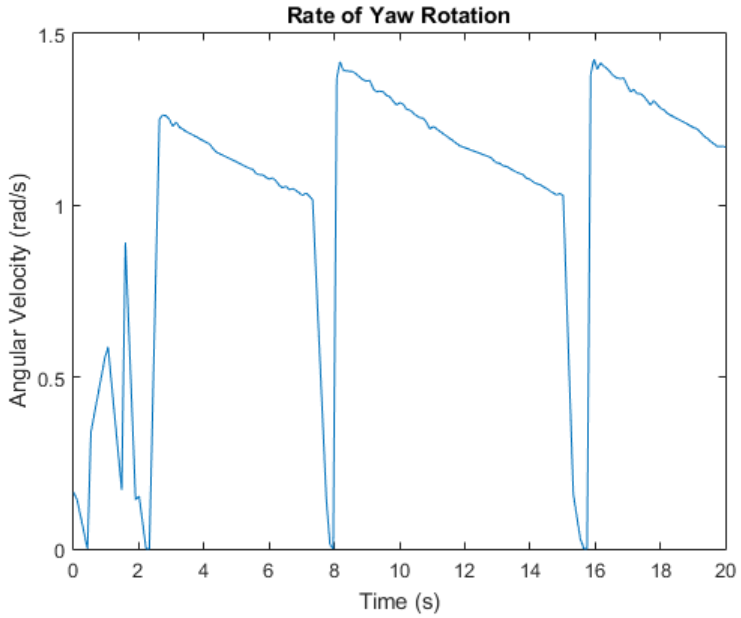

Figure 4.13: TableSat yaw rotation rate during spin up

Observing the rigid body's rate of rotation, shown in Fig. 4.13, it can be seen that the rigid body overshoots the desired rate of rotation (1.047 radians per second) and then slows down via air resistance and friction.It is also noted that the values decrease to zero each time the thrusters fire. It is the author's opinion that this is due to magnetic interference from the solenoid valves opening each time the thrusters fire. While this does not adversely affect the TableSat's performance, it does result in less-than-clean looking data especially during the transient phase of the spin up when the thrusters are constantly firing. Looking at the boom deflection data, it can be seen that this data is particularly noisy. The deflections in the boom during spin up conditions are so small that the strain cannot measure them amongst the noise. Though some large spikes in the data exist, it is clear that the median point of the boom deflection is at zero.

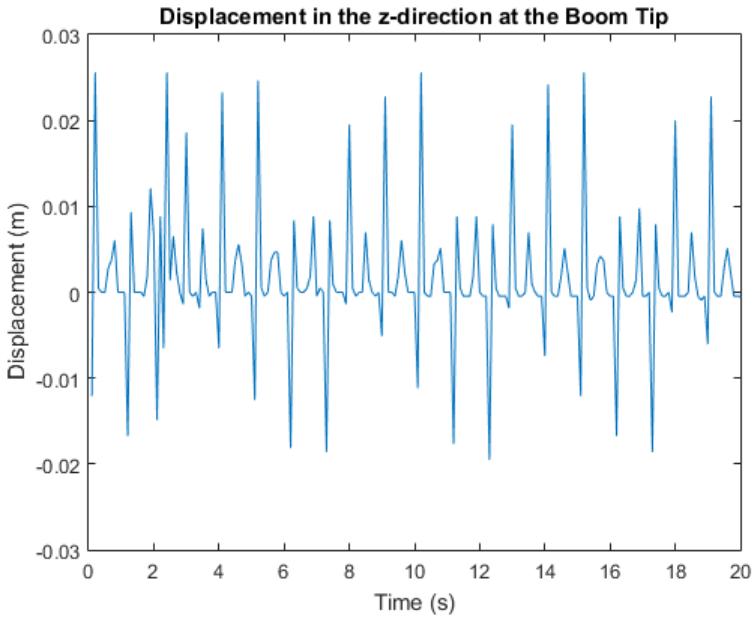

Figure 4.14: Displacement of the boom tip in the z-direction during spin up

4.5.4 Steady State Disturbance

In this experiment, a disturbance is applied to the tip of a boom once the TableSat reaches steady state. The LQR controller's ability to compensate for the disturbance is the point of investigation. The rate of rotation of the rigid body and concurrent boom deflection are shown in Fig. 4.15 and Fig. 4.16. The missing portion of the signal is due to the effects of saturation of the Arduino's analog-to-digital converter. It is

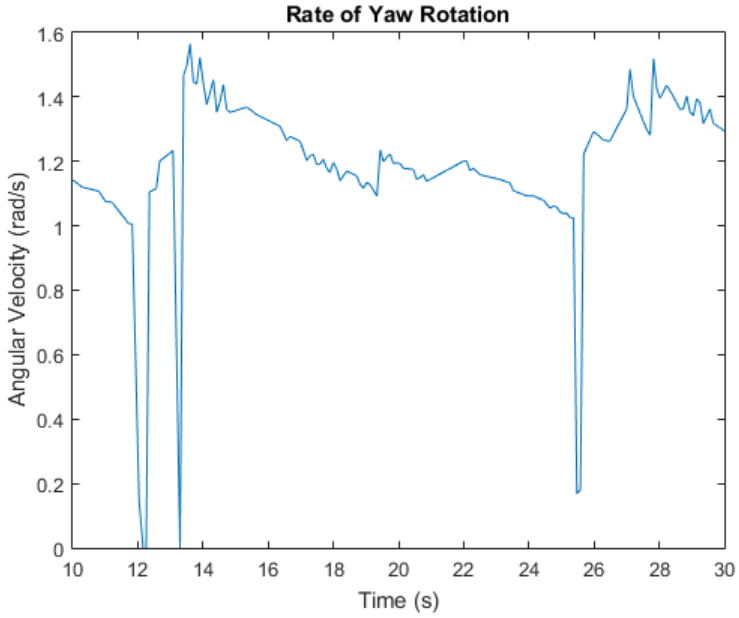

Figure 4.15: TableSat yaw rotation rate after a disturbance during steady state

noted that the results of this section start at ten seconds and not at zero seconds. This is to ensure that the TableSat is well into its steady state phase before applying the

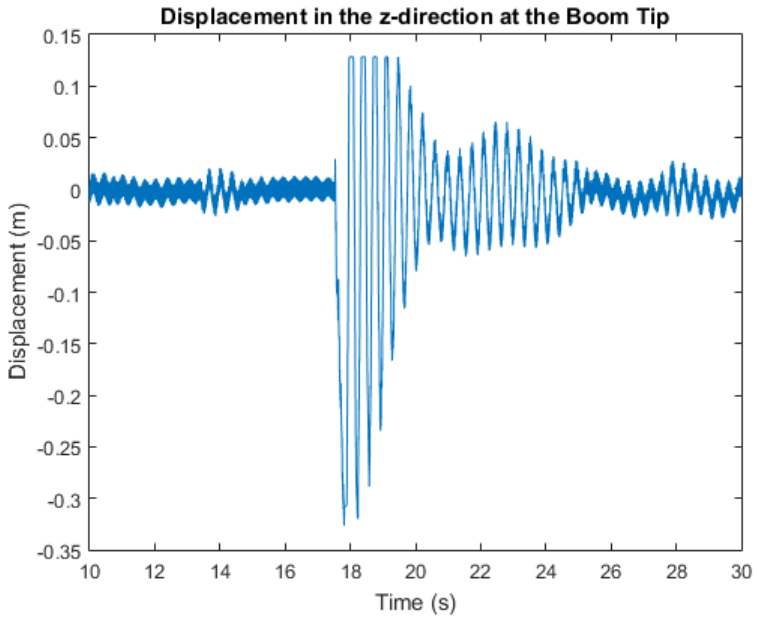

Figure 4.16: Displacement of the boom tip in the z-direction after a disturbance during steady state

disturbance. The transient phase (from zero to ten seconds) is removed from the plot to make the data more clear. Looking first at the boom deflection data in Fig. 4.16, the larger oscillations are saturated due to the Arduino's analog-to-digital converter. The reason the saturation is not symmetrical on the bottom of the signal is because the zero point of the strain gauge sensor is not in the middle of the Arduino's analog-to-digital converter's range. Despite this, the boom deflection data shows the same trend as the impulse force data. Looking at the corresponding rigid body rotation data, shown in Fig. 4.15, one might not recognize when exactly the disturbance takes place. Noting from Fig. 4.16 that the disturbance takes place at about 17s and looking at the corresponding 17 second mark in this figure, one can see the slight oscillations during the decreasing speed of the TableSat after it overshoots its desired rate of rotation. Based on this, it would seem that a disturbance to the tip of a boom does not have a significant effect on the rigid body (unless very large in magnitude). This hypothesis is investigated in the modified hybrid algorithm equivalent of this experiment.

Chapter 5

Modified Algorithm Simulations

In this chapter, simulations of the modified hybrid algorithm are presented. First, simulations with similar conditions as the previous chapter are presented for comparison with and validation from the results from experiments with the TableSat. These conditions include: constant external moment on the rigid body, impulse force on a boom, spinning up from rest, and steady state disturbance. Next, a simulation is presented as a prediction of the dynamics of the true MMS s/c. This simulation is the true MMS s/c spinning up from rest to a desired spin rate. It should also be noted that although the x-axis of these plots is not measured in time directly, it is measured in iterations of the algorithm which are one hundredth of a second each. For example, the 220th iteration is such that $t = 2.2s$.

5.1 Simulations for Experimental Validation

5.1.1 Constant External Moment

In this simulation, a constant moment is applied onto the rigid body from the thrusters. The applied moment is in the positive z-direction (rigid body's frame). No controllers are active in this simulation. From Fig. 5.1, it can be seen that the angular velocity of the rigid body reaches steady state once the effects of drag and friction reach equilibrium with the constant external moment. It is because of this simulation that the corrective factor for the drag/friction is added. Before the corrective factor was added, the angular velocity of the rigid body would increase without reasonable bounds because the drag force on the system (modeled in Chapter 2) was so insignificant relative to the external moment. Though the air bearing provides nearly frictionless rotation, friction does exist and was previously unmodeled. To correct this, an empirically determined factor of 1.9×10^5 is multiplied to the modeled drag force to account for the unmodeled drag and friction. This allows the drag force to hinder the model from approaching unreasonably high angular velocities while remaining insignificant at lower angular velocities. The deflections of the boom at Node 2 and Node 3 are shown in Fig. 5.2 and Fig. 5.3, respectively. These figures show the boom deflection increasing with some oscillations as the angular velocity increases and then reaching a steady state constant deflection as the angular velocity also reaches steady state. Fig. 5.4 shows a very slight change in the system moment of inertia. Because the TableSat's booms are not very long and do not undergo large deflections, it is anticipated that the mass moment of inertia of the system would not vary greatly for this model. This

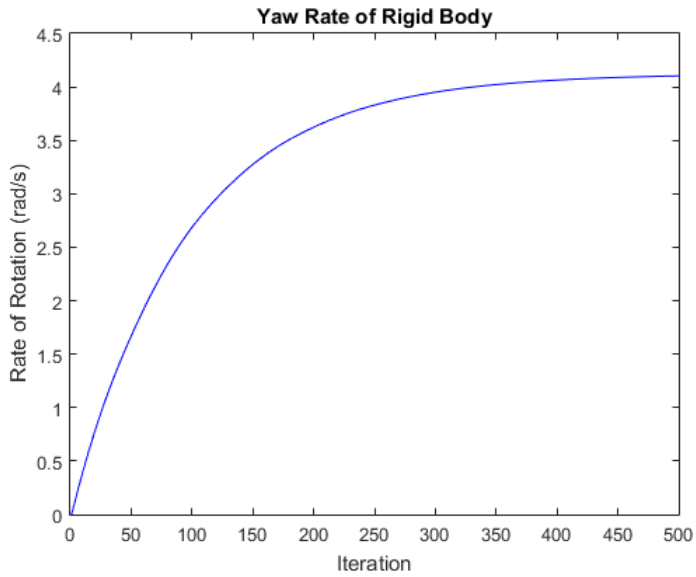

Figure 5.1: Simulated yaw rotation rate of the rigid body during constant external moment

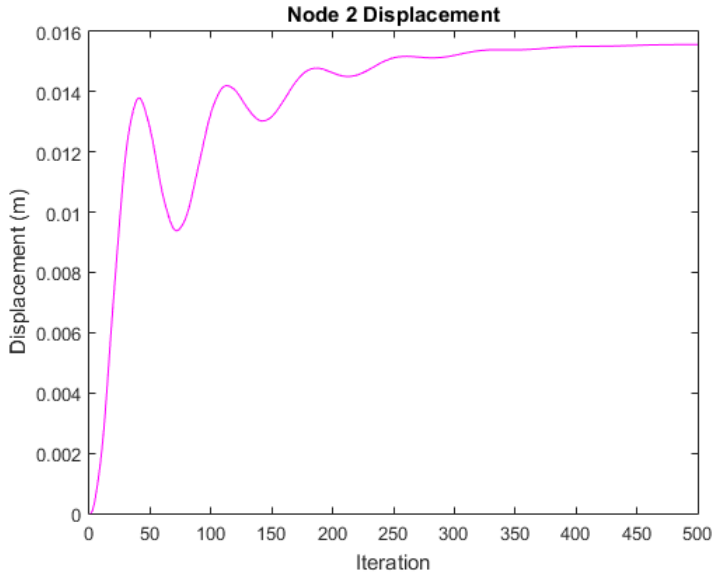

Figure 5.2: Boom displacement at Node 2 in z-direction during constant external moment

thought is confirmed in this and the following simulations.

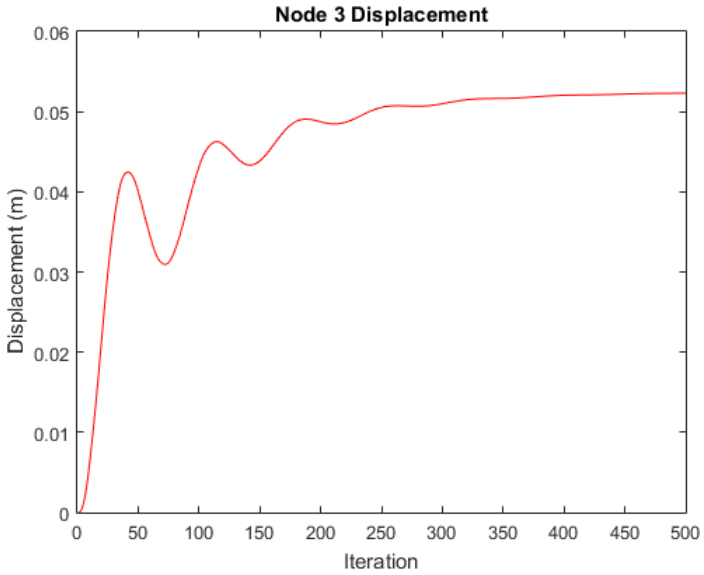

Figure 5.3: Boom displacement at Node 3 in z-direction during constant external moment

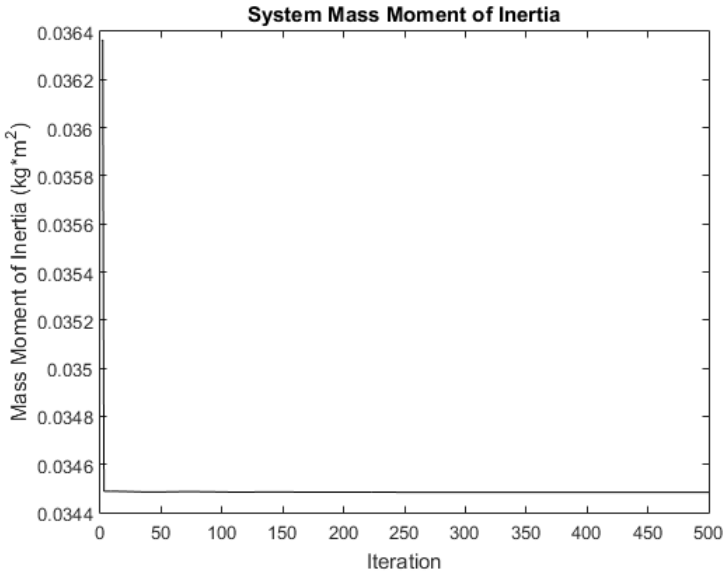

Figure 5.4: Time-varying mass moment of inertia during constant external moment

5.1.2 Impulse Force and Free Response

In this simulation, an external impulse force of 100 Newtons is placed on the tip of a boom in the positive z-direction (boom's frame) and the resulting free response of the boom and rigid body is shown. No controllers are active in this simulation. Looking at Fig. 5.6 and Fig. 5.7, the impulse force on the boom tip occurs at the 200th iteration (at $t = 2s$). The resulting magnitude of deflection is about twice as large as for the constant external moment experiment. This results in a greater variation in the system mass moment of inertia, shown in Fig. 5.8. As the booms oscillate and damp out to zero deflection, the mass moment of inertia oscillates as well before damping out to its steady state value. Fig. 5.5 shows how the motion of the boom is propagated to the

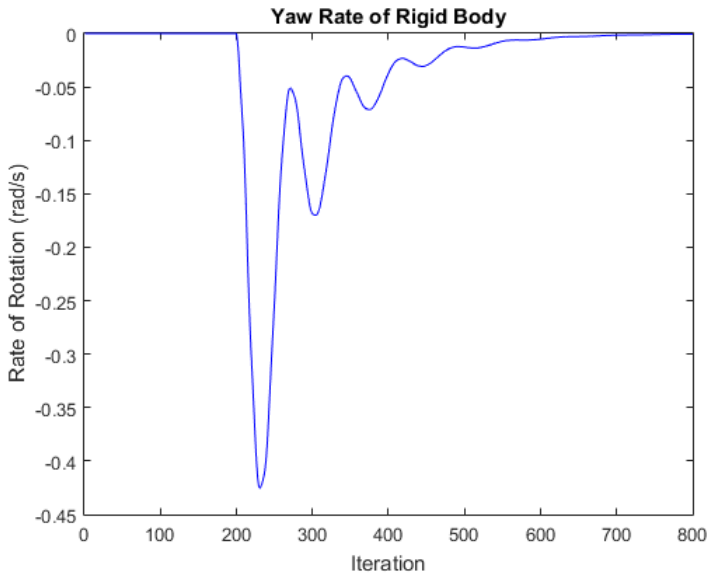

Figure 5.5: Simulated yaw rotation rate of the rigid body during free response after impulse force on boom

rigid body. Once the boom is initially deflected, there is a spike in the angular velocity of the rigid body which decays back to zero velocity during the free response. It is important to note that in the simulation, the booms do not appear to excite motion within one another.

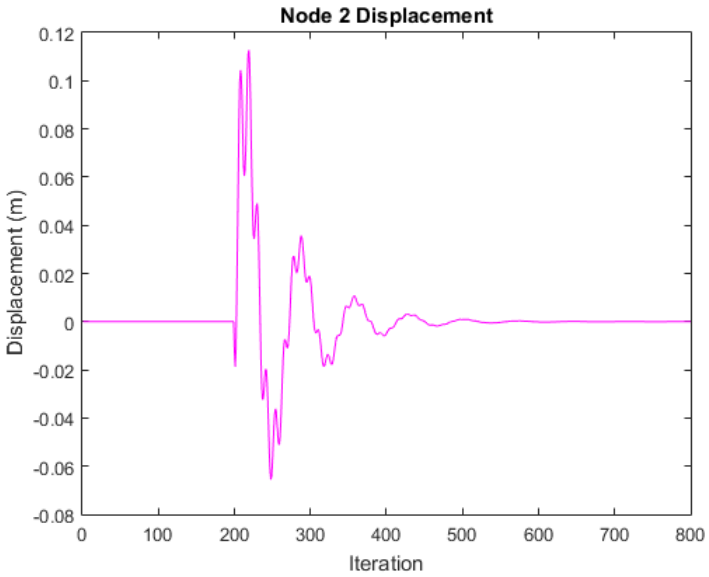

Figure 5.6: Boom displacement at Node 2 in z-direction during free response after impulse force on boom

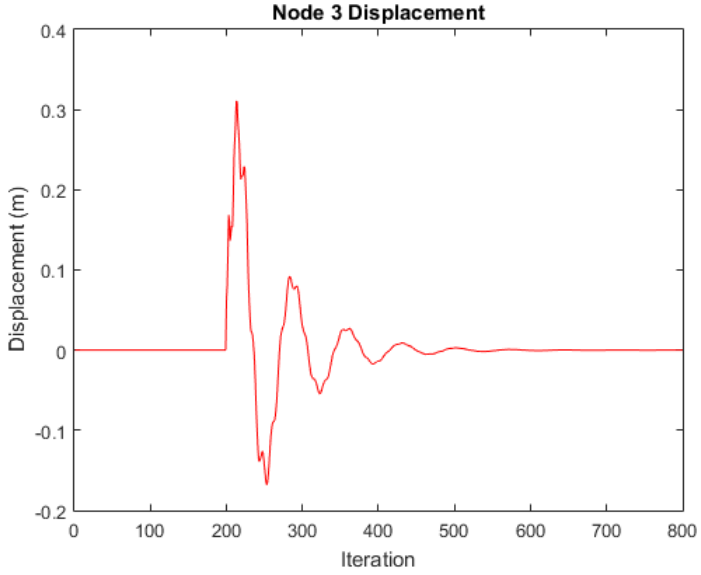

Figure 5.7: Boom displacement at Node 3 in z-direction during free response after impulse force on boom

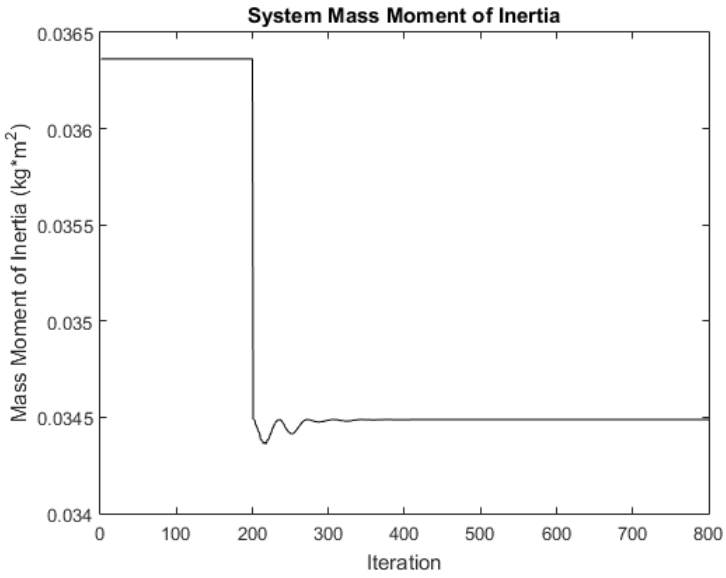

Figure 5.8: Time-varying mass moment of inertia during free response after impulse force on boom

5.1.3 Spin Up from Rest

In this simulation, the s/c rigid body and booms start at rest and with zero displacement and spins up to a desired spin (yaw) rate of three radians per second. The results using each of the three controllers are presented. Because of the conditions of this simulation, the only results that concern us are that of the yaw rotation rate, the displacement of the booms in the spin-plane (z-direction in the boom's frame), and the mass moment of inertia about the z-axis (rigid body's frame).

PID Controller

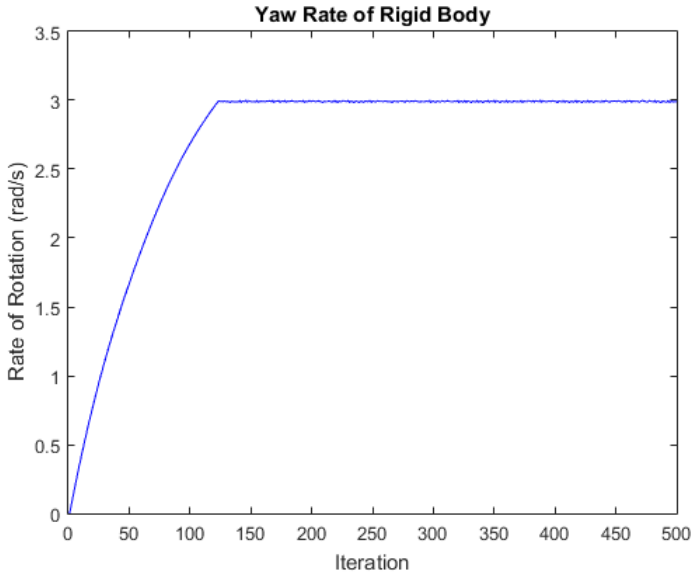

Figure 5.9: Simulated yaw rotation rate of the rigid body during spin up using PID controller

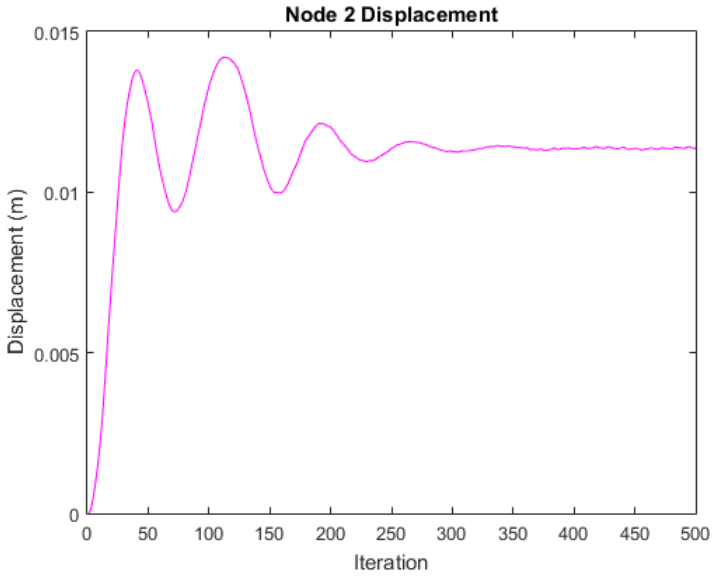

Figure 5.10: Boom displacement at Node 2 in z-direction during spin up using PID controller

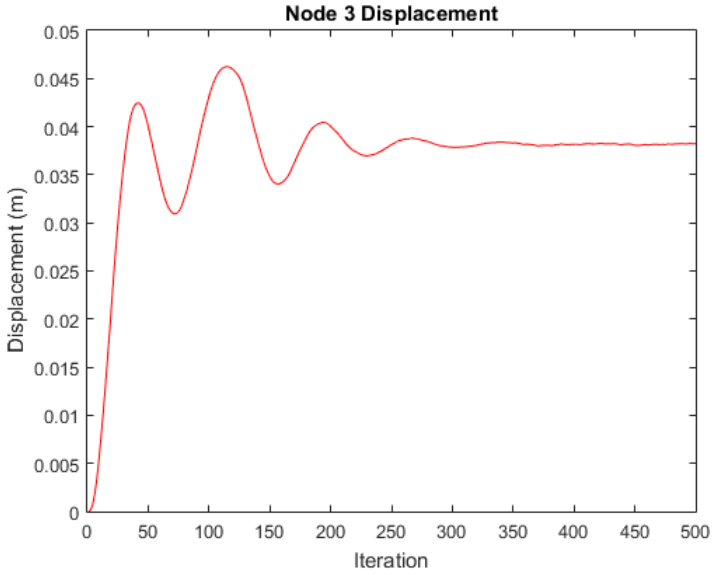

Figure 5.11: Boom displacement at Node 3 in z-direction during spin up using PID controller

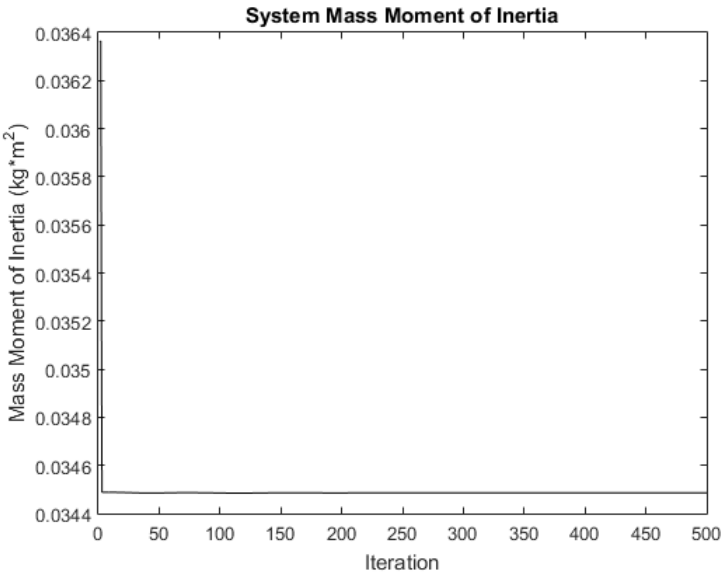

Figure 5.12: Time-varying mass moment of inertia during spin up using PID controller

SMC Controller

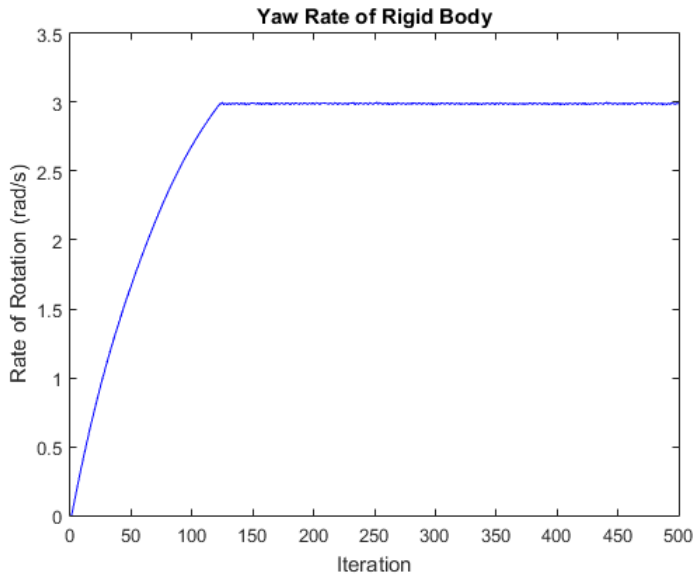

Figure 5.13: Simulated yaw rotation rate of the rigid body during spin up using SMC controller

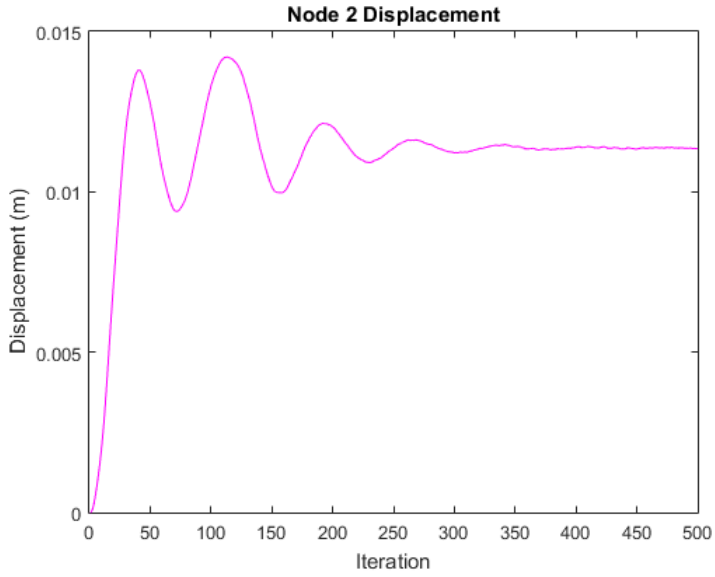

Figure 5.14: Boom displacement at Node 2 in z-direction during spin up using SMC controller

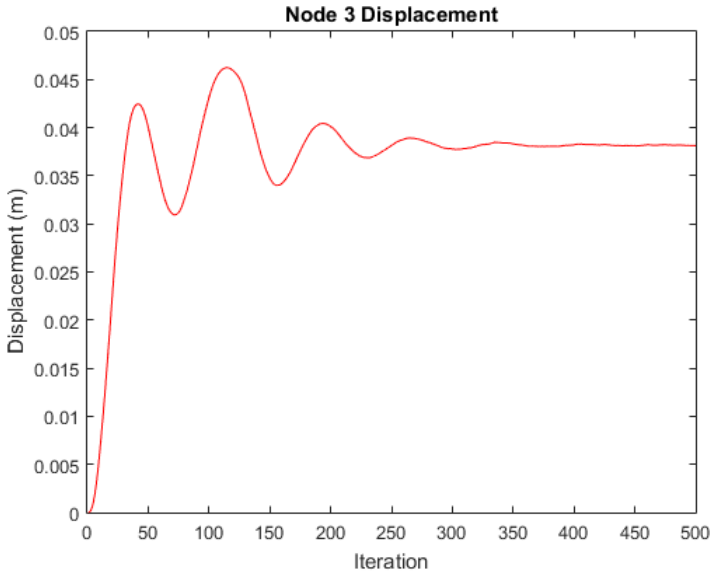

Figure 5.15: Boom displacement at Node 3 in z-direction during spin up using SMC controller

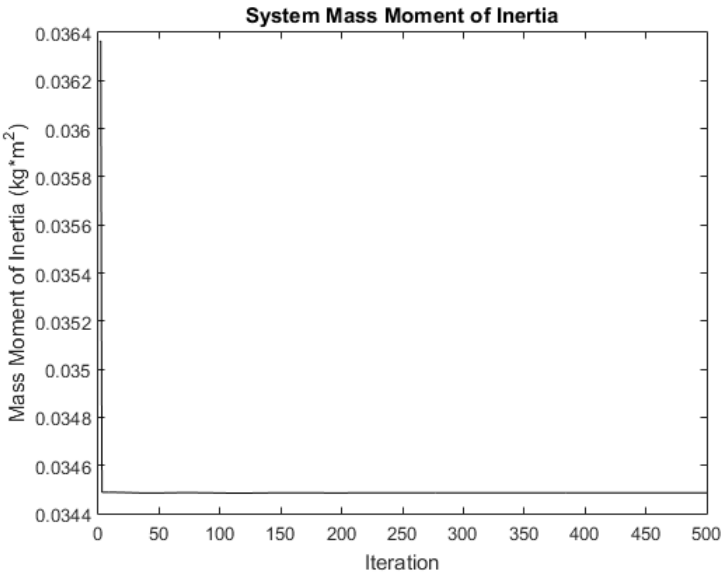

Figure 5.16: Time-varying mass moment of inertia during spin up using SMC controller

LQR Controller

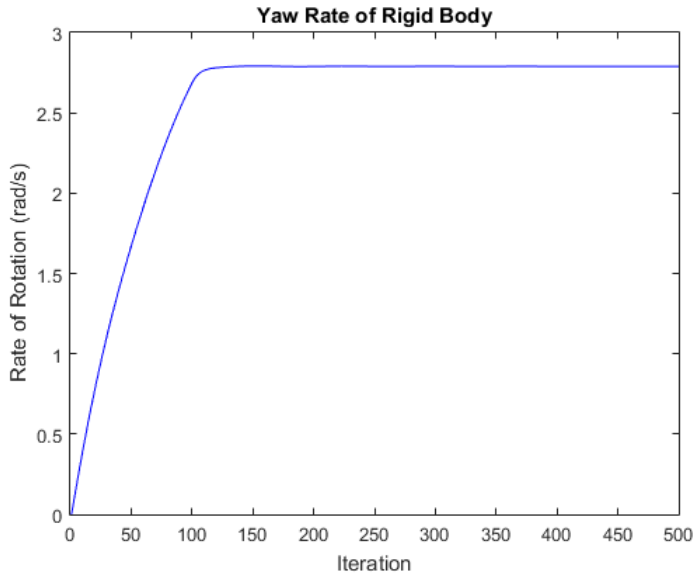

Figure 5.17: Simulated yaw rotation rate of the rigid body during spin up using LQR controller

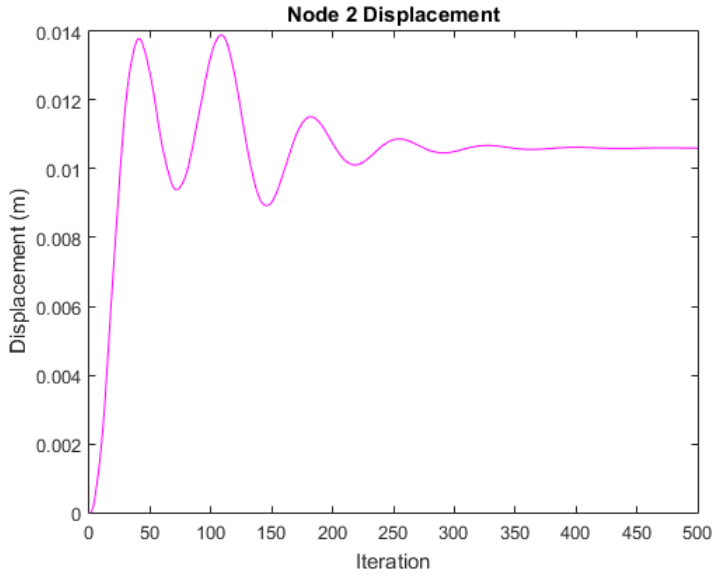

Figure 5.18: Boom displacement at Node 2 in z-direction during spin up using LQR controller

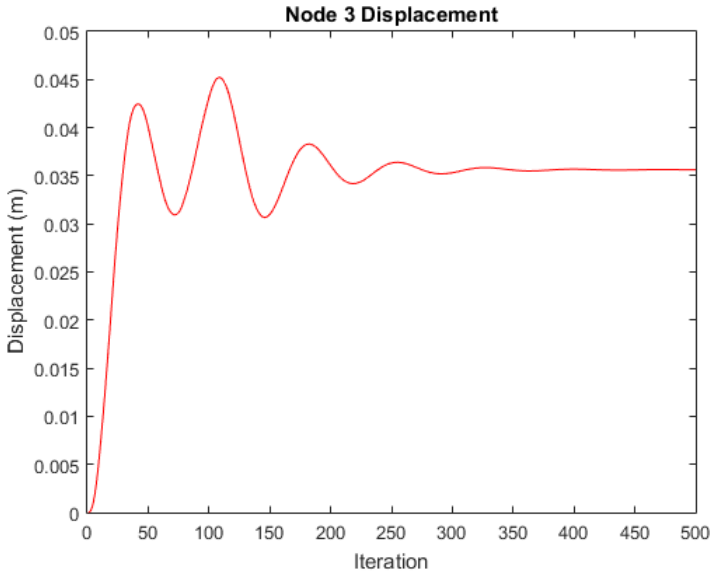

Figure 5.19: Boom displacement at Node 3 in z-direction during spin up using LQR controller

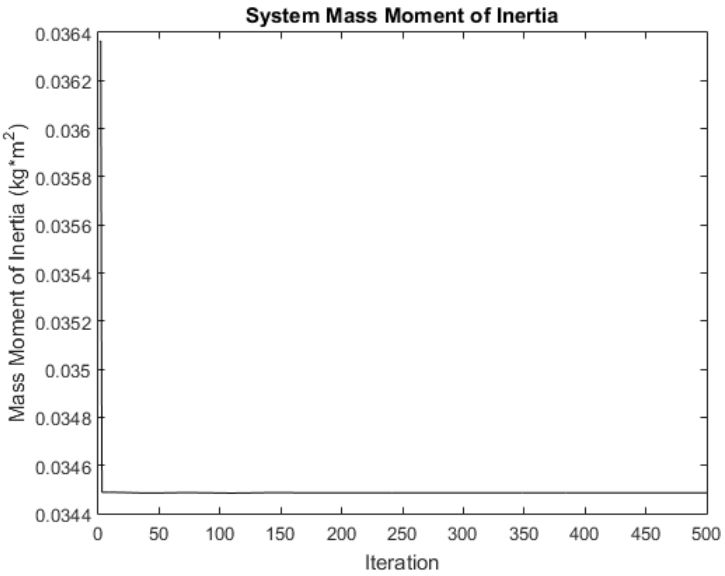

Figure 5.20: Time-varying mass moment of inertia during spin up using LQR controller

This simulation is the first to compare the three different controllers and their performance. Due to the relatively low angular velocity of the rigid body in this experiment, the boom deflections are not so large as to result in a significantly varying mass moment of inertia, as previously mentioned. The mass moment of inertia of the system under each of the PID, SMC, and LQR controllers is shown in Fig. 5.12, Fig. 5.16, and Fig. 5.20, respectively. The main points of interest of this simulation are the ability keep a desired spin rate and boom deflection. The PID and SMC controllers have nearly identical responses for both the desired spin rate (Fig. 5.9 and Fig. 5.13, respectively) and boom deflection (Fig. 5.10 and Fig. 5.11, Fig. 5.14 and Fig. 5.15, respectively). These controllers approach the desired spin rate quickly and with no steady state error. The boom deflections have their initial spike while they are still under the force of the thrusters, then decay to a steady state deflection once the thrusters stop constantly firing at maximum thrust. The spin rate under the LQR controller (shown in Fig. 5.17) does not approach its steady state value as quickly as the other controllers. When it does reach steady state, there is significant steady state error. The boom deflection for this case is shown in Fig. 5.18 and Fig. 5.19. Because the thrusters are not firing as much with this controller, the oscillations of the booms are not as large as with the other two cases.

5.1.4 Steady State Disturbance

In this simulation, a disturbance is applied once the system reaches steady state. In the first scenario, the disturbance is applied to the boom tip in the form of an impulse force, much like the previously mentioned TableSat test run. In the second scenario, the disturbance is applied to the rigid body in the form of a step function. Two disturbances are tested in each scenario. The first disturbance is greater than the maximum force the thrusters can output. The second disturbance is at a level less than the maximum force the thrusters can output. All three controllers are compared in the simulations.

PID Controller: Boom Disturbance

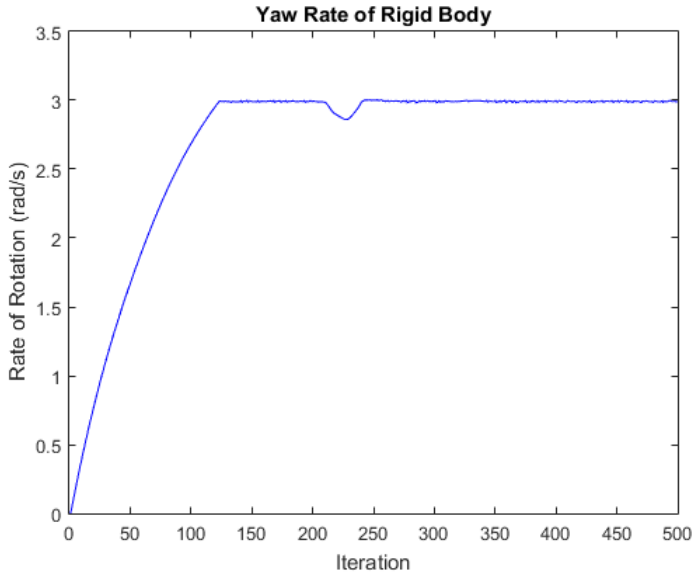

Figure 5.21: Simulated yaw rotation rate of the rigid body with PID and boom disturbance

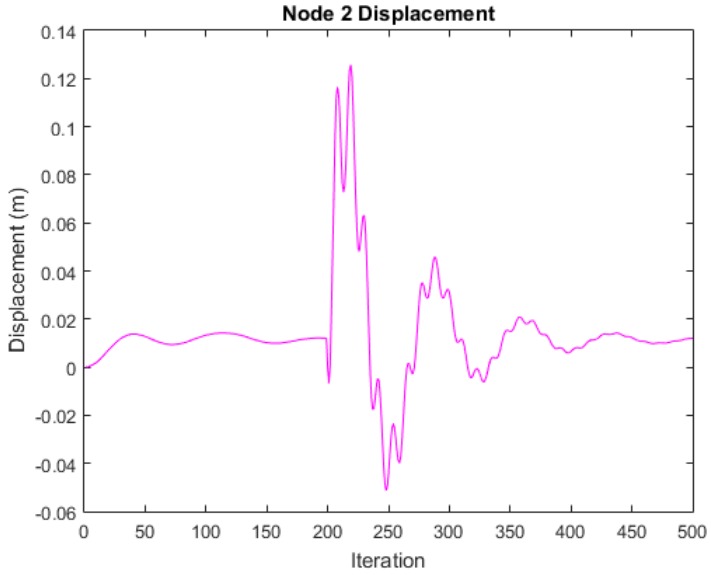

Figure 5.22: Boom displacement at Node 2 in z-direction with PID and boom disturbance

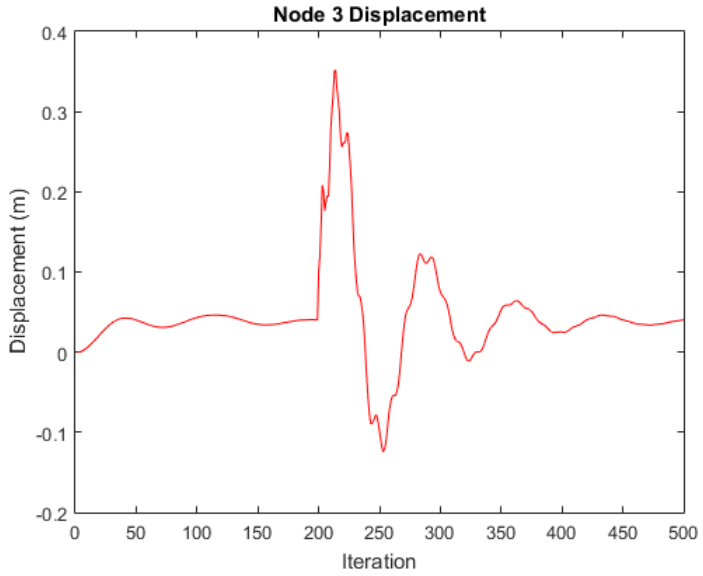

Figure 5.23: Boom displacement at Node 3 in z-direction with PID and boom disturbance

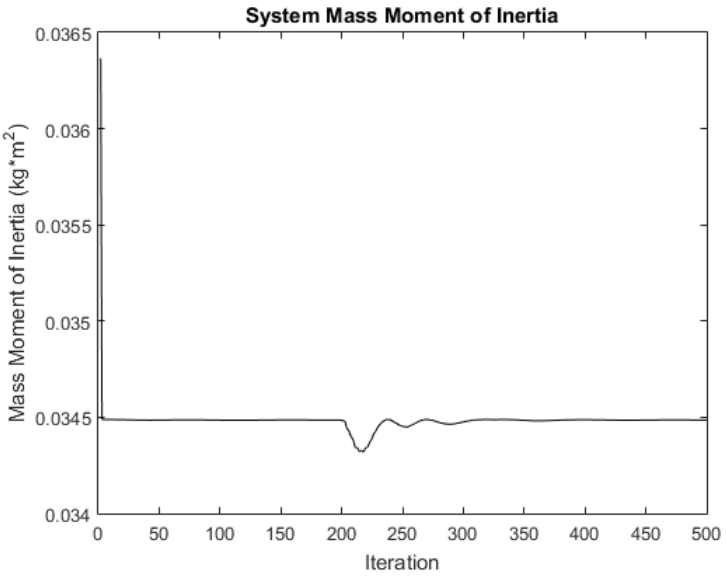

Figure 5.24: Time-varying mass moment of inertia with PID and boom disturbance

SMC Controller: Boom Disturbance

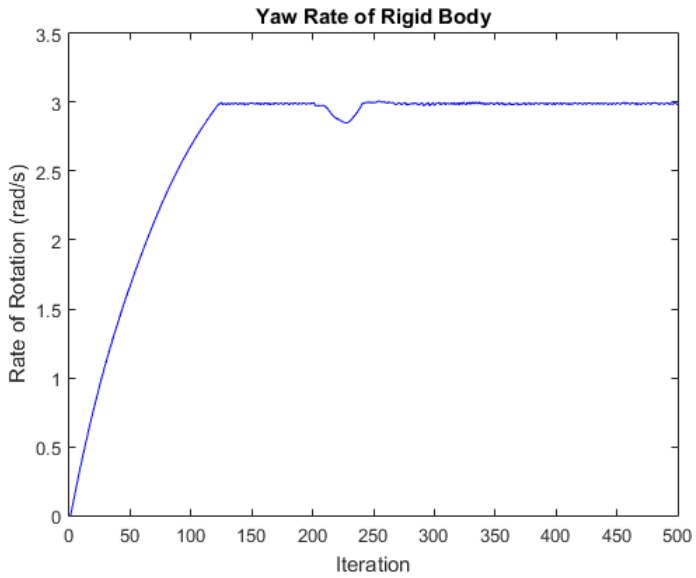

Figure 5.25: Simulated yaw rotation rate of the rigid body with SMC and boom disturbance

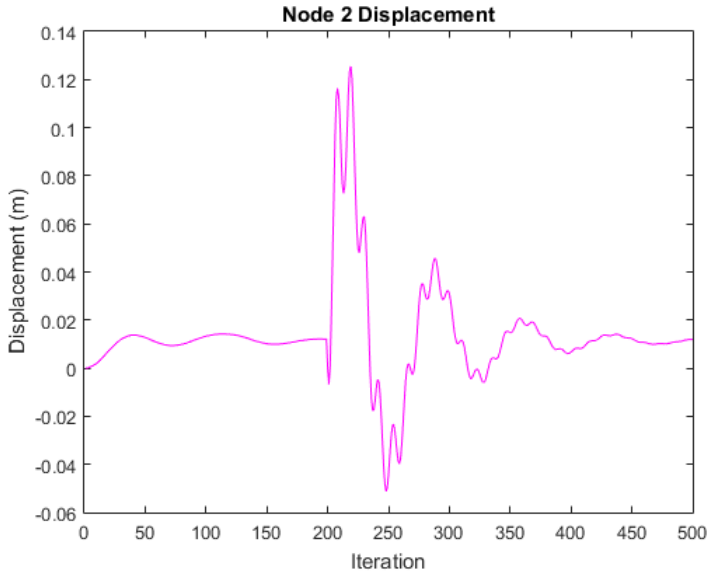

Figure 5.26: Boom displacement at Node 2 in z-direction with SMC and boom disturbance

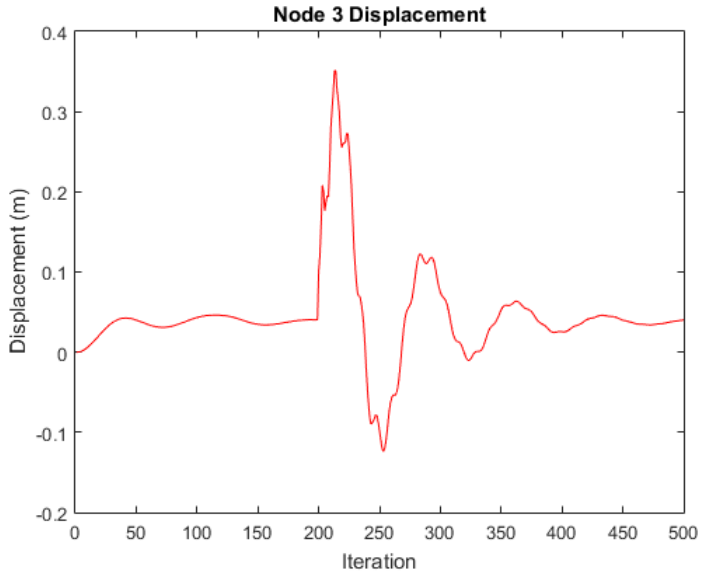

Figure 5.27: Boom displacement at Node 3 in z-direction with SMC and boom disturbance

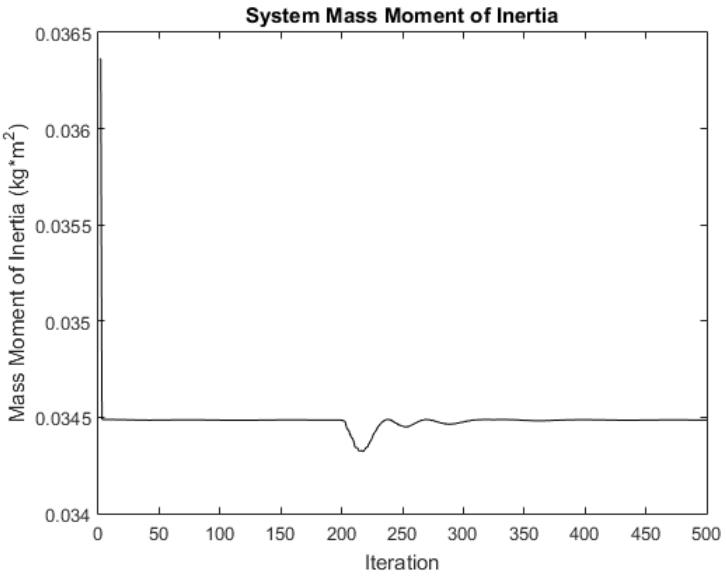

Figure 5.28: Time-varying mass moment of inertia with SMC and boom disturbance

LQR Controller: Boom Disturbance

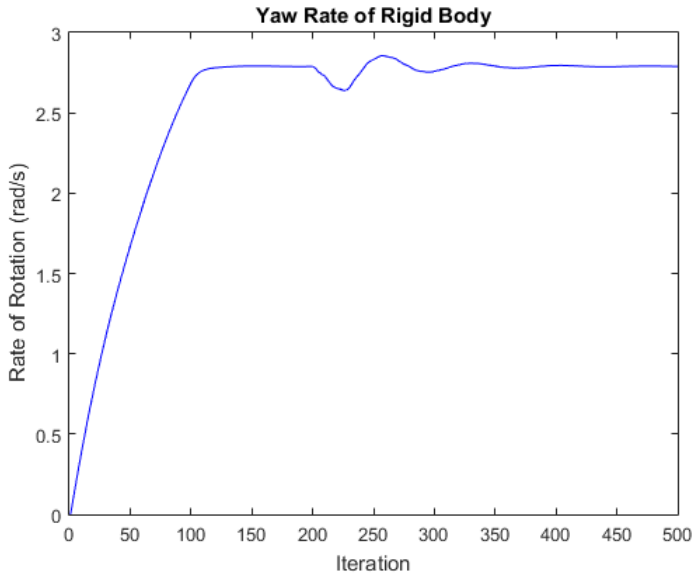

Figure 5.29: Simulated yaw rotation rate of the rigid body with LQR and boom disturbance

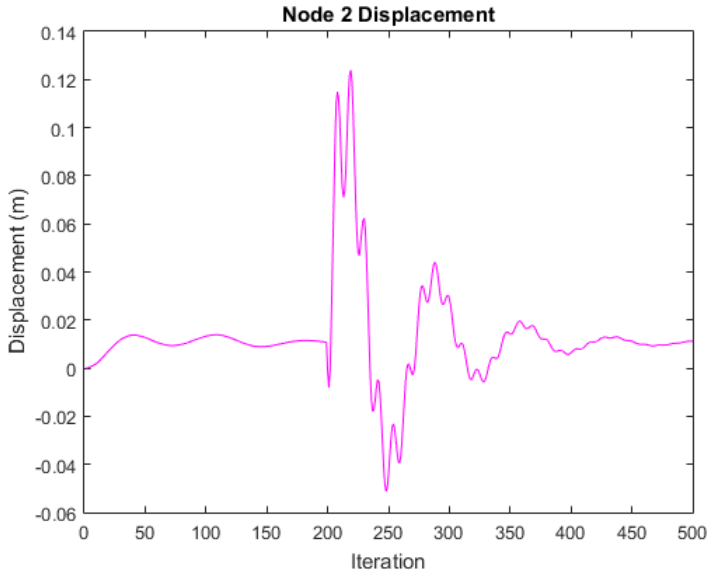

Figure 5.30: Boom displacement at Node 2 in z-direction with LQR and boom disturbance

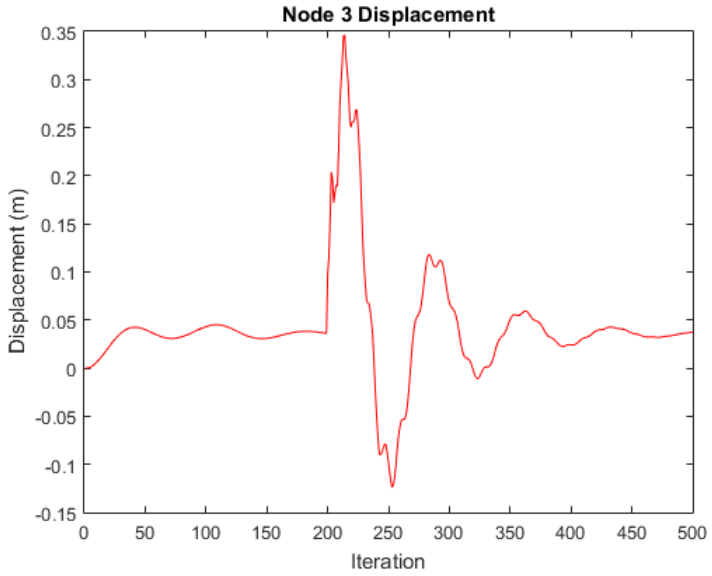

Figure 5.31: Boom displacement at Node 3 in z-direction with LQR and boom disturbance

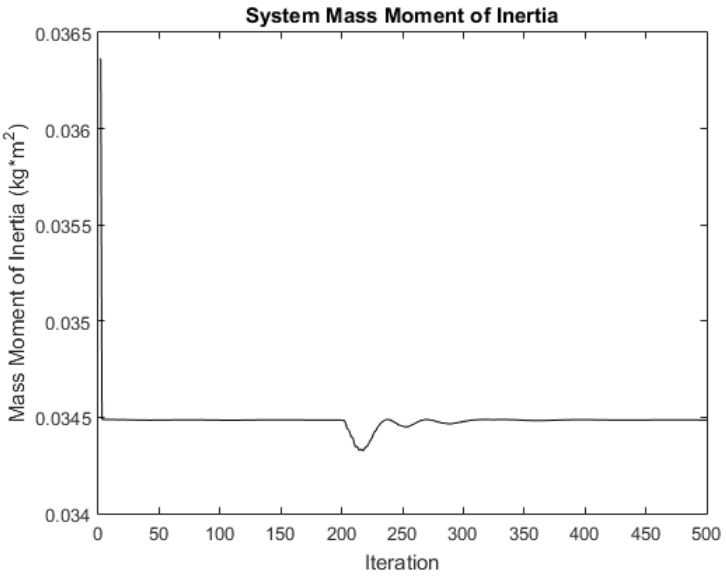

Figure 5.32: Time-varying mass moment of inertia with LQR and boom disturbance

Observing the boom deflections first (Fig. 5.22 and Fig. 5.23 for PID, Fig. 5.26 and Fig. 5.27 for SMC, and Fig. 5.30 and Fig. 5.31 for LQR), the magnitude by which the boom is initially displaced and the resulting response from the booms is shown. In all three cases, the booms have vibrations within the oscillations from the impact of the force. At the moment the disturbance is applied (iteration 200, $t = 2s$), the three controllers have very different responses in regards to the spin rate, however. The PID and SMC controllers show good performance of rejecting the disturbance with minimal oscillations, shown in Fig. 5.21 and Fig. 5.25, respectively. The LQR controller, however, takes about 200 iterations (two seconds) to recover from the disturbance and still has steady state error. The mass moment of inertia for the PID, SMC, and LQR controllers, shown in Fig. 5.24, Fig. 5.28, and Fig. 5.28, respectively. There is very little variance in these plots.

PID Controller: Rigid Body Disturbance

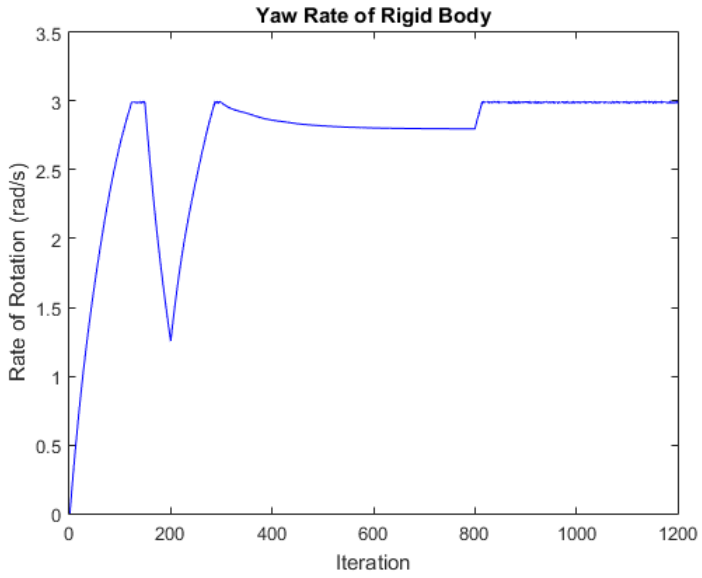

Figure 5.33: Simulated yaw rotation rate of the rigid body with PID and rigid body disturbances

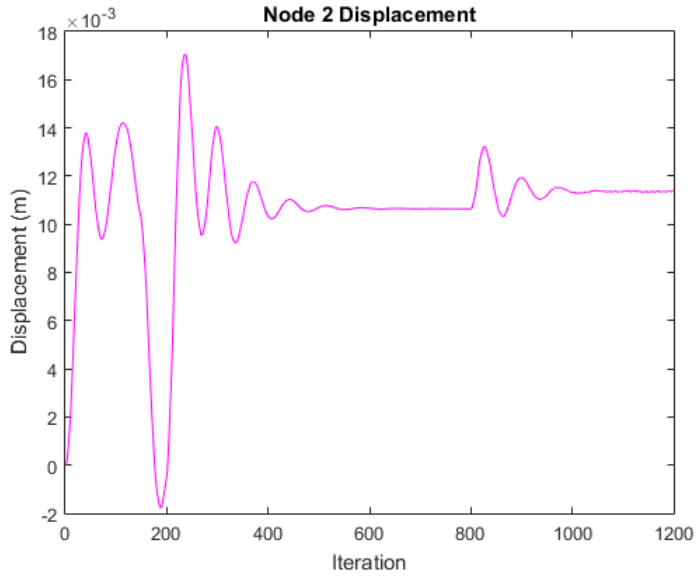

Figure 5.34: Boom displacement at Node 2 in z-direction with PID and rigid body disturbances

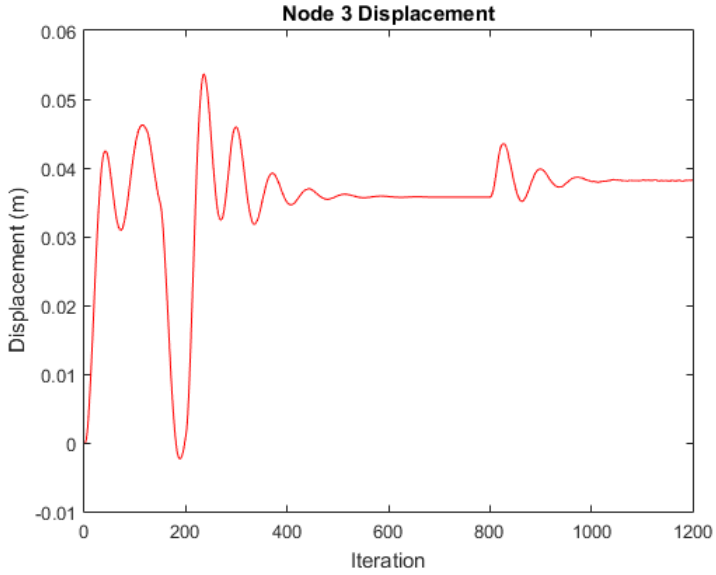

Figure 5.35: Boom displacement at Node 3 in z-direction with PID and rigid body disturbances

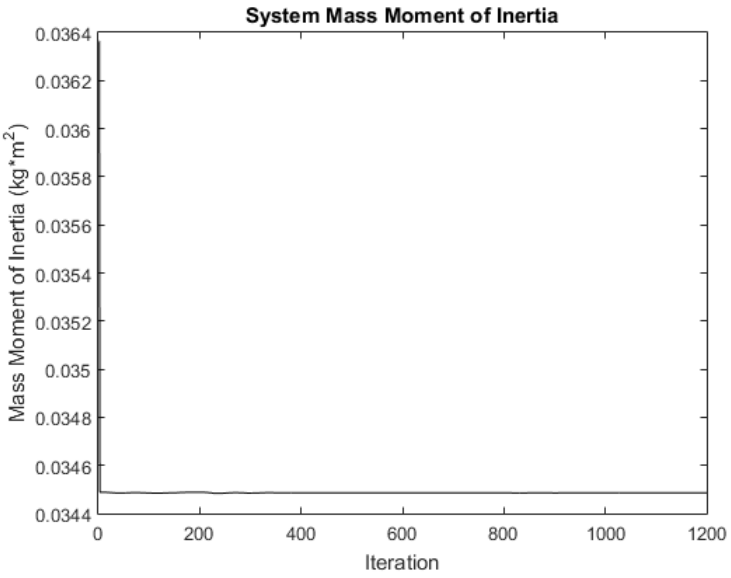

Figure 5.36: Time-varying mass moment of inertia with PID and rigid body disturbances

SMC Controller: Rigid Body Disturbance

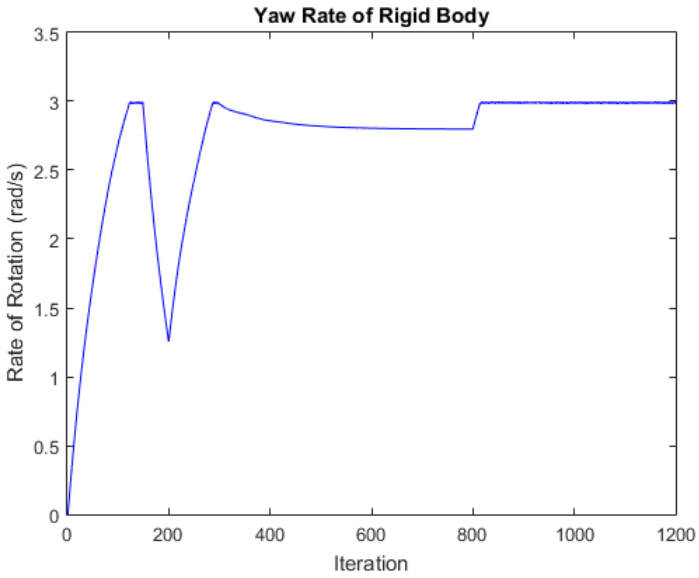

Figure 5.37: Simulated yaw rotation rate of the rigid body with SMC and rigid body disturbances

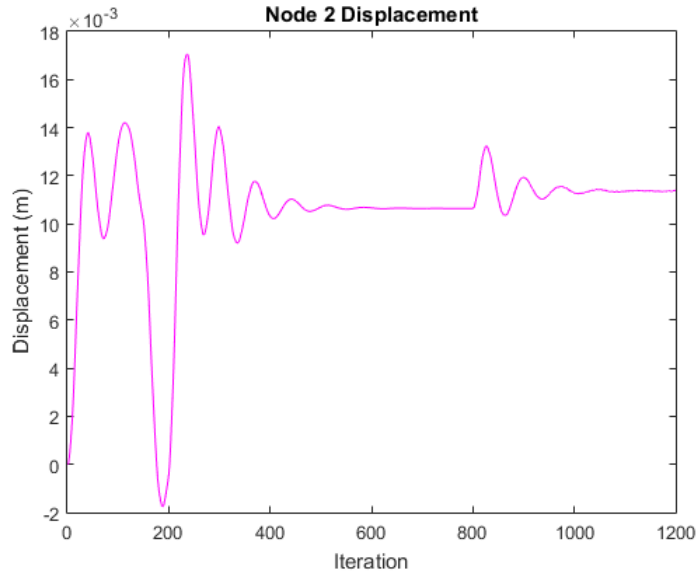

Figure 5.38: Boom displacement at Node 2 in z-direction with SMC and rigid body disturbances

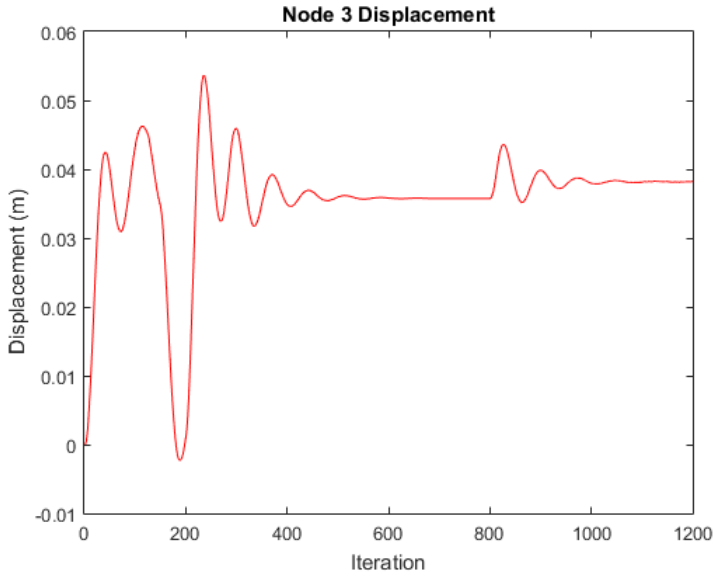

Figure 5.39: Boom displacement at Node 3 in z-direction with SMC and rigid body disturbances

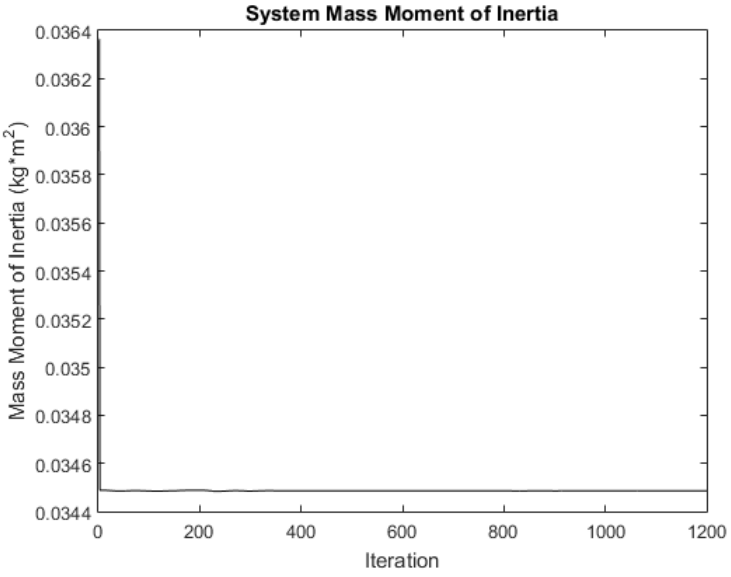

Figure 5.40: Time-varying mass moment of inertia with SMC and rigid body disturbances

LQR Controller: Rigid Body Disturbance

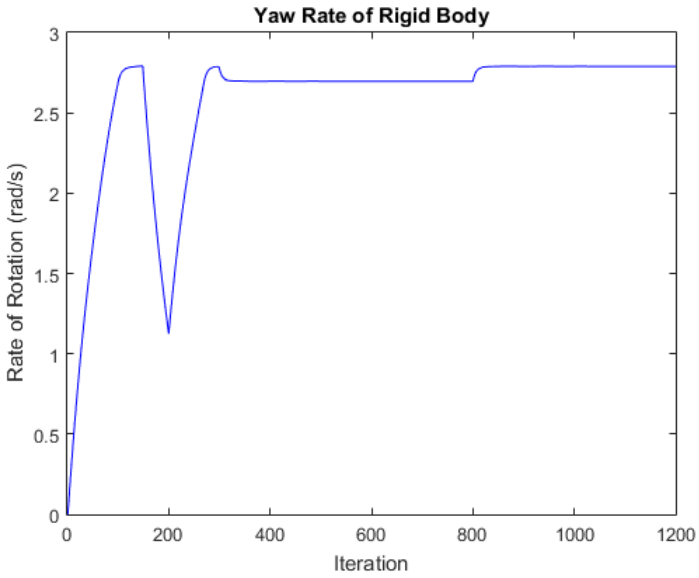

Figure 5.41: Simulated yaw rotation rate of the rigid body with LQR and rigid body disturbances

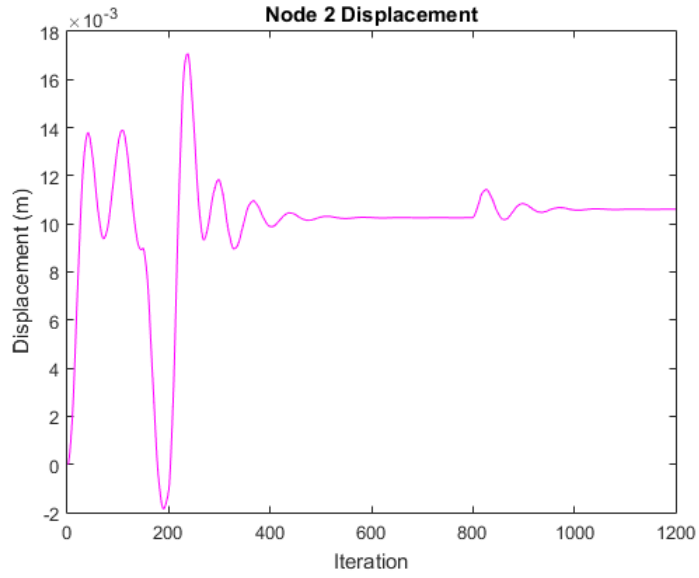

Figure 5.42: Boom displacement at Node 2 in z-direction with LQR and rigid body disturbances

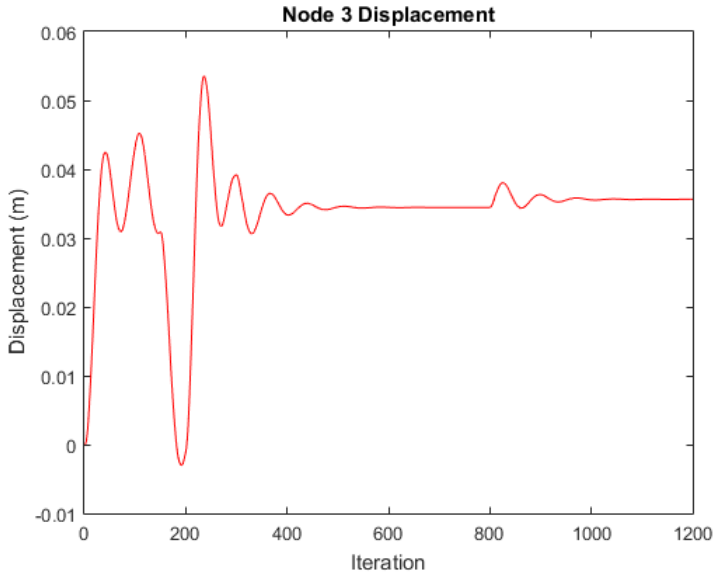

Figure 5.43: Boom displacement at Node 3 in z-direction with LQR and rigid body disturbances

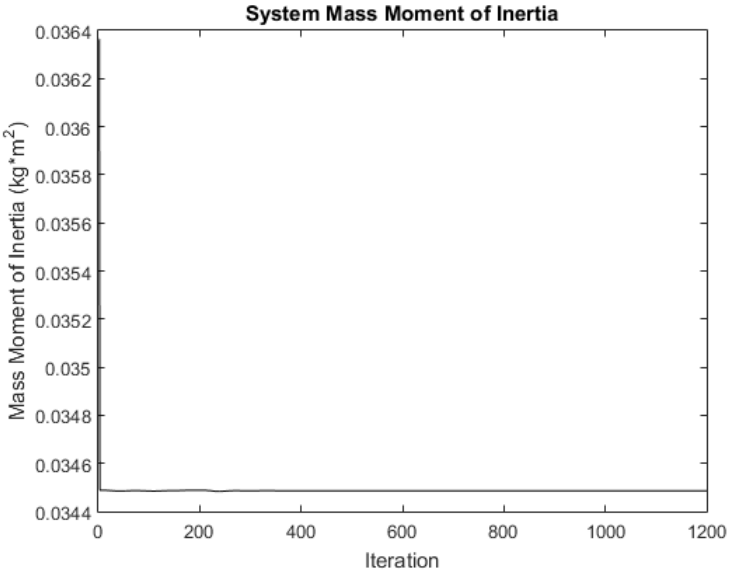

Figure 5.44: Time-varying mass moment of inertia with LQR and rigid body disturbances

In the PID and SMC controller cases, the first disturbance overpowers the thrusters during its duration and begins to decrease the spin rate, as shown in Fig. 5.33 and Fig. 5.37, respectively. Once the first disturbance ends, the spin rate recovers quickly. When the second disturbance is applied, both controllers prevent the disturbance from affecting the spin rate of the rigid body significantly. Both disturbances have a significant effect on the boom deflection, as shown in Fig. 5.34, Fig. 5.35, Fig. 5.38, and Fig. 5.39. The LQR controller handles the first disturbance in much of the same way, however, it is less affected by the second disturbance at steady state, shown in Fig. 5.41. Since there is still a significant camount of steady state error with the LQR controller however, it ends up being no better than the PID or SMC controllers. The boom deflections with the LQR controller, shown in Fig. 5.42 and Fig. 5.43, are very similar to the boom deflections for the PID and SMC controllers. The mass moment of inertia of the system does not vary greatly in all three cases, as shown in Fig. 5.36, Fig. 5.40, and Fig. 5.44.

5.2 Prediction Simulations

5.2.1 Spin Up from Rest

In this simulation, the modified hybrid algorithm is used to simulate the true MMS satellite spinning up from rest to a desired 0.314 radians per second (3 rpm) using the PID controller. The MMS hub has mass $m_{hub} = 1360kg$, radius $r_{hub} = 3.4m$, and height $h_{hub} = 1.2m$ [20]. The booms have diameter $d_{boom} = 0.00155m$ and density $\rho_{boom} = 0.00506\frac{kg}{m}$ [21]. Since the booms of the true MMS s/c are braided wire booms, the elastic modulus is assumed to be that of copper, $E = 117 \times 10^9 Pa$.

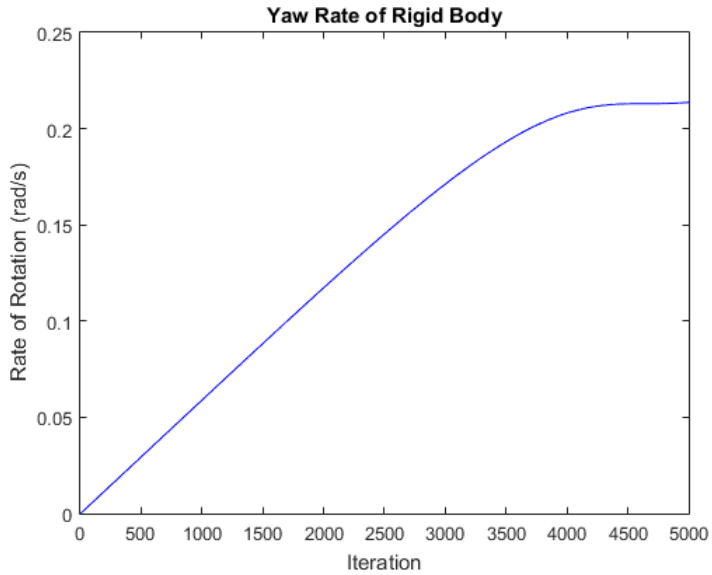

Figure 5.45: Simulated yaw rotation rate of the rigid body under true MMS conditions

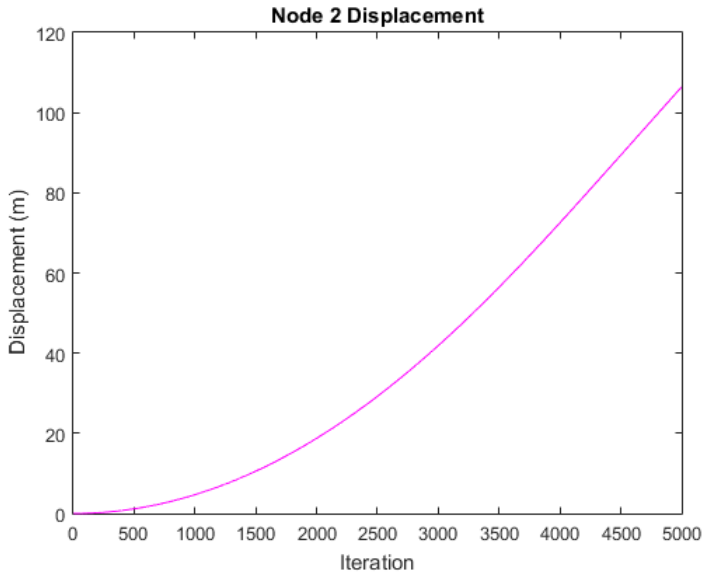

Figure 5.46: Boom displacement at Node 2 in z-direction under true MMS conditions

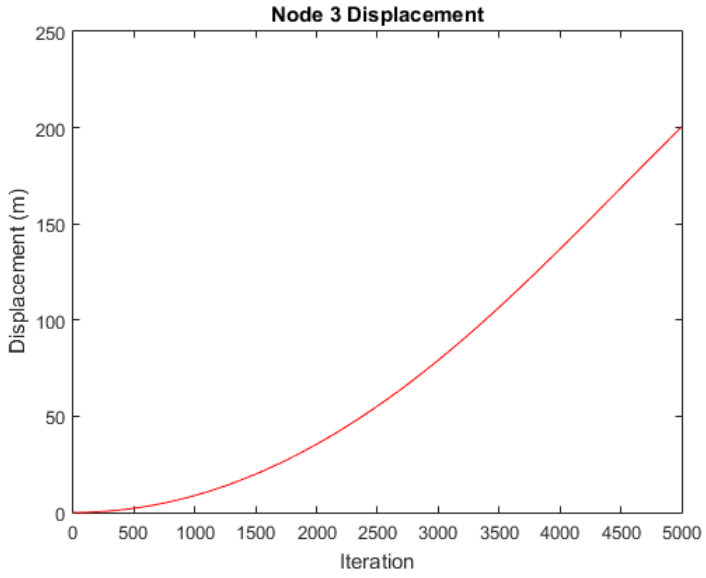

Figure 5.47: Boom displacement at Node 3 in z-direction under true MMS conditions

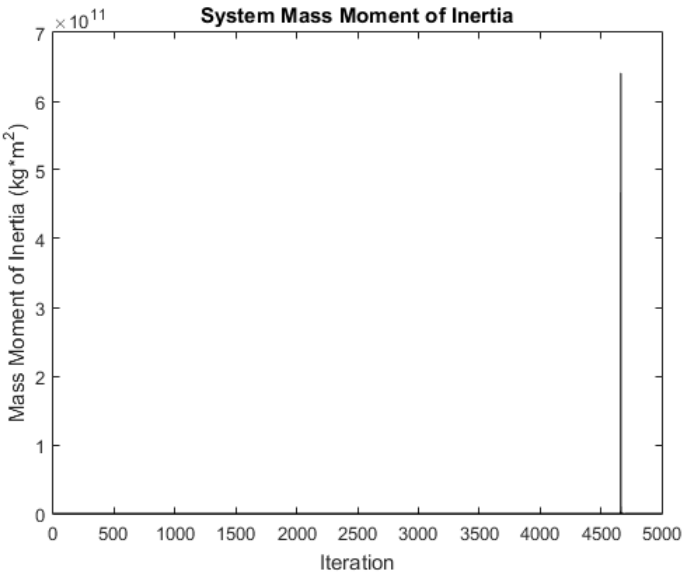

Figure 5.48: Time-varying mass moment of inertia under true MMS conditions

Chapter 6

Discussion of Results

In this chapter, the results from the two previous chapters are compared in order to assess the validity of the hybrid algorithm. First, the results of the constant external moment and impulse force from both the TableSat and the modified hybrid algorithm are compared. The frequency of the boom oscillations in both the TableSat and modified hybrid algorithm results is also analyzed. Next, the performances of the three controllers will be analyzed based on the results from the spin up and disturbance rejection simulations. Lastly, the results from the prediction simulations of the modified hybrid algorithm under MMS conditions will be discussed in detail.

6.1 Comparison

The general trend when comparing the results from the TableSat and the modified hybrid algorithm is that the latter produces results that are accurate, at the very least, to the same order of magnitude as the results of the TableSat. The constant external moment and free response simulations are verification that the results from the modified hybrid algorithm are reasonably accurate. Conveniently, the former simulation provides a case where the rigid body motion is propagated to the booms and the latter simulation provides a case where the boom motion is propagated to the rigid body. In these simulations, the results exhibit similar behavior to those of the TableSat. While the magnitudes of the rate of rotation and boom deflection are not exact, they are well within acceptable standards. Table 6.1 shows some of the results between the TableSat and modified hybrid algorithm side-by-side. The results are not perfect, but

Test	Metric (Maximum)	TableSat	Mod. Hybrid Algorithm
Const. Moment	Rate of Rotation	4.1 rad/s	4.1 rad/s
	Node 3 Deflection	0.02 meters	0.052 meters
Free Response	Rate of Rotation	0.1 rad/s	0.4 rad/s
	Node 3 Deflection	0.15 meters	0.3 meters

Table 6.1: Comparison of results from TableSat and modified hybrid algorithm

there are several areas of improvement for the modified hybrid algorithm which are discussed in Chapter 7.

It is noted that the TableSat's booms have a much greater frequency of oscillation than the booms simulated by the modified hybrid algorithm. The most likely reason for

this is the vibrations in the boom. Because the strain gauge sensor can only measure one mode of deflection in the boom, deflection in other modes (such as vibrations) can interfere with the measurements of the strain gauge sensor. The modified hybrid algorithm can model multiple modes of deflection, as seen by the coupled oscillations in Fig. 5.7. The limitation of the strain gauges to only measuring one mode of deflection results in a skewing of the frequency of oscillation.

6.2 Controller Performance

Based on the presented results, it can clearly be seen that under the current conditions, the LQR falls short of meeting the requirements of the experiments. It has a significant steady state error and takes longer to reject disturbances than the PID or SMC controller. The LQR controller also, however, uses the least amount of fuel, as the reader can see by comparing the thruster outputs of the LQR, PID, and SMC controllers during the boom disturbance simulation, shown in Fig. 6.1, Fig. 6.2, and Fig. 6.3, respectively. This means that the Q and R matrices can be redefined to allow for more state accuracy at the cost of fuel consumption. In order to decide between

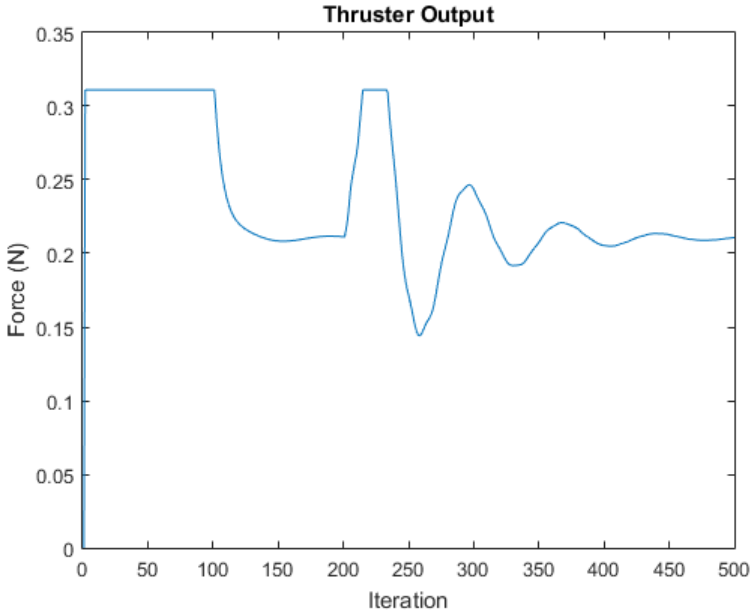

Figure 6.1: Thruster output of the system during the boom disturbance simulation using the LQR controller

the PID and SMC controllers, which performed similarly well reaching the desired spin rate and rejecting disturbances, the amount of thruster fuel consumed must be investigated. Using a numerical computation software to calculate the area under the curves in Fig. 6.2 and Fig. 6.3, it is found that the PID controller consumes slightly less fuel than the SMC controller to perform the same task with the same accuracy. Repeating this calculation with the other simulations in which the controllers were active yields similar results. This larger consumption of fuel by the SMC controller is due to the effects of chattering, as previously mentioned. Thus, it can be said that the PID controller is the better choice for this application.

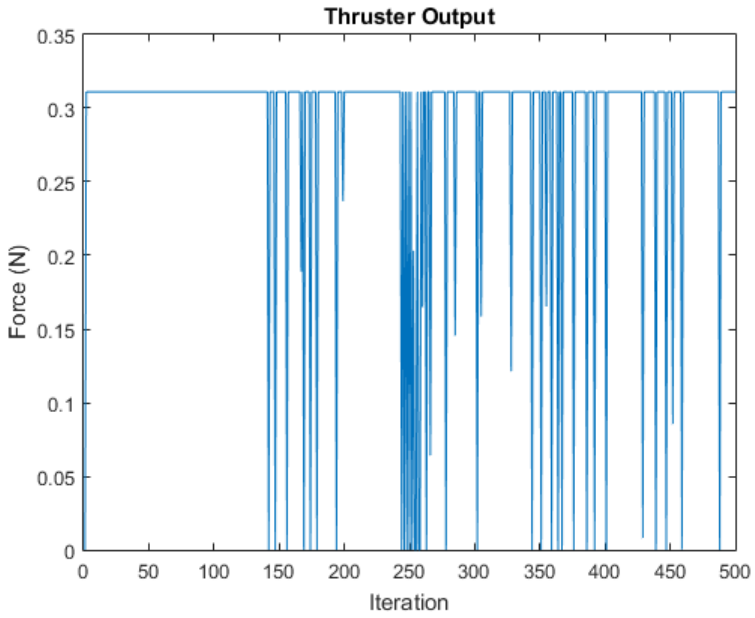

Figure 6.2: Thruster output of the system during the boom disturbance simulation using the PID controller

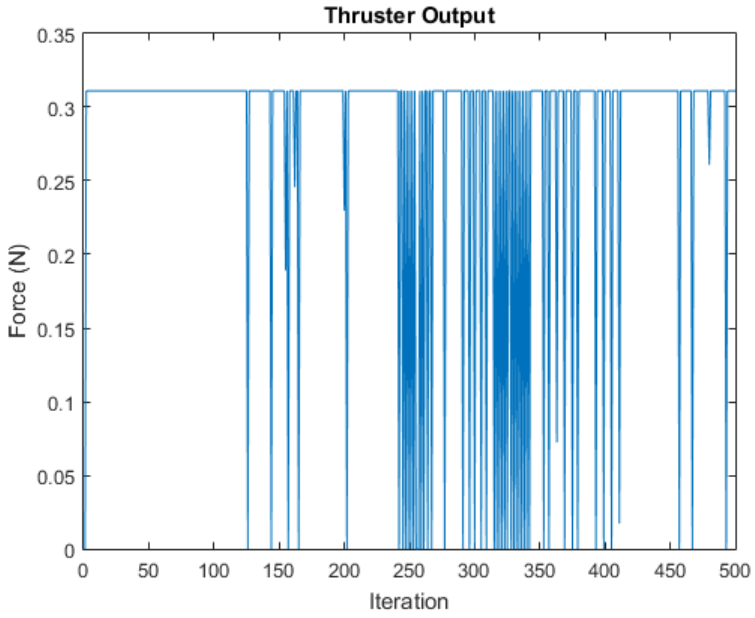

Figure 6.3: Thruster output of the system during the boom disturbance simulation using the SMC controller

6.3 Prediction Simulations

It only takes the one simulation of the true MMS s/c using the modified hybrid algorithm to realize that the algorithm does not perform as expected. The problem arises from the method by which the FEM is implemented, which assumes a linear elastic response. The FEM produces boom deflections that are much larger than even the booms length. These very large deflections skew the mass moment of inertia of the booms, creating numerical singularities in the moment of inertia. In turn, the s/c

cannot reach its desired spin rate due to the extremely high moment of inertia. The large deformation experienced by the booms due to their large length act like a non-linearity that the finite element procedure, using the current implementation, cannot accurately model.

Chapter 7

Conclusions and Future Work

There are three main objectives of this research:
1. Develop a self-contained, computationally efficient, proof-of-concept mathematical model for the MMS s/c.
2. Prove the modified hybrid algorithm is accurate and reliable by experimentally validating it using the TableSat test platform.
3. Apply various controllers including PID, SMC, and LQR controllers to the mathematical model to investigate which controller performs best against disturbances to the system.

The proposed mathematical model utilizes both Euler's Moment Equations for the rigid body dynamics and finite element analysis for the flexible structure dynamics. These two methods are interfaced to create a modified hybrid algorithm that addresses the issues with Medas and Thein's hybrid algorithm.

The algorithm begins by propagating the rigid body dynamics using Euler's Moment Equations and then using the rotation information to calculate the boundary conditions that drive the finite element model. The finite element model then runs a user-defined number of times before outputting a boom displacement vector. This displacement vector is then used to update the mass moment of inertia of the system and calculate the reaction moment applied on the rigid body from the booms before the algorithm proceeds to the next iteration, starting again with Euler's Moment Equations. Several different simulations are run using the modified hybrid algorithm for verification with the TableSat IC test platform.

The TableSat is retrofitted with strain gauges instead of accelerometers for measuring the deflection of the booms. These strain gauges prove to be far more reliable and accurate than the accelerometers, though they still have their own issues. Several tests are run on the TableSat and data is gathered that confirms the validity of the modified hybrid modeling algorithm.

When assessing which of the three controllers is the most effective for this application, the best choice is dependent on desired goals. If fuel consumption is a low priority, then the Sliding Mode Controller or PID controller are the clear choices. These controllers exhibit the most robust disturbance rejection. If fuel consumption is a high priority, the LQR controller is the best choice because the cost function can be optimized to prioritize fuel consumption. Although there is steady state error, the Q and R matrices can be manipulated to find a trade off between fuel consumption and state accuracy. The PID controller is slightly better than SMC in the sense that it does not consume as much fuel, but is just as accurate. The largest concern with

the PID controller in this application is that if the system were to stray too far from the linearization point, the controller could cause the system to become unstable.

The modified hybrid algorithm is then used to simulate true MMS conditions. However, the algorithm does not handle the large deformations experienced by the booms under MMS conditions well and proves to be largely inaccurate. This is due to the method by which the FEM is implemented, which assumes a linear elastic response. This assumption holds true for the TableSat model, but not for true MMS conditions. To rectify this, a FEM that accounts for the geometrically nonlinear booms must be developed.

There are a few different areas of improvement for future research. Where the FEM is concerned, the current modified hybrid algorithm uses a two element system to model the booms. Increasing the number of elements per boom increases the computational accuracy of the FEM but would also increase the computation time of the algorithm as a whole. Further investigation would be required to determine a reasonable trade off between FEM accuracy and computation time. Another area of improvement lies in the moment of inertia calculation. The current algorithm uses a highly simplified method for calculating the moment of inertia. Almost certainly, a more in-depth investigation into calculating the mass moment of inertia of irregular shapes would make the algorithm more accurate and more stable. When propagating the rigid body dynamics, a simplified version of Euler's Moment Equations are used that only utilize the principal axes of the mass moment of inertia tensor. Using the full mass moment of inertia tensor would make the Euler Moment Equations more accurate when propagating the rigid body dynamics. In regards to the experimental platform, the strain gauge sensors, though accurate at times, could also be inconsistent as is evident from the asymmetrical saturation of th Arduino's analog-to-digital converter. More research into the amplifier in the strain gauge circuit would required to make these sensors more reliable. Any future work on this topic should prioritize improvements as listed in the order presented here.

Lastly, generalizing this modified hybrid algorithm for any flexible spin-stabilized s/c would be possible, as long as sufficient knowledge of the geometric and material properties is known and defined a priori. The FEM matrices must be redefined for changes in geometry of the flexible structures and, therefore, the boundary conditions that drive the FEM must also be rederived for the new geometry.

Bibliography

[1] "Magnetospheric Multiscale (MMS) Mission." *The Magnetospheric Multiscale (MMS) Mission.* NASA, n.d. Web. 15 Aug. 2016.

[2] *"New NASA Goddard Video: MMS Mission Overview."* SciTech Daily. N.p., 24 Feb. 2015. Web. 15 Aug. 2016.

[3] Eric Stoneking. "Newton-Euler Dynamic Equations of Motion for a Multi-Body Spacecraft", AIAA Guidance, Navigation and Control Conference and Exhibit, Guidance, Navigation, and Control and Co-located Conferences

[4] Jason Medas and May-Win Thein. "Hybrid Modeling Technique Applied to NASA's Magnetospheric MultiScale (MMS) Mission TableSat Generation IB (TableSat IB)", AIAA/AAS Astrodynamics Specialist Conference, Guidance, Navigation, and Control and Co-located Conferences

[5] Kaplan, Marshall H. Modern Spacecraft Dynamics & Control. New York: Wiley, 1976. Print.

[6] "FEM for Frames (Finite Element Method) Part 1." What-when-how. N.p., n.d. Web. 16 Aug. 2016.

[7] Bathe, Klaus-Jurgen, and Klaus-Jurgen Bathe. Finite Element Procedures. N.p.: n.p., 2006. Print.

[8] Hibbeler, R. C. Mechanics of Materials. N.p.: n.p., n.d. Print.

[9] Munson, Bruce Roy, Donald F. Young, and T. H. Okiishi. Fundamentals of Fluid Mechanics. Hoboken, NJ: J. Wiley & Sons, 2006. Print.

[10] Nise, Norman S. Control Systems Engineering. Hoboken, NJ: Wiley, 2011. Print.

[11] Slotine, J.-J E., and Weiping Li. Applied Nonlinear Control. Englewood Cliffs, NJ: Prentice Hall, 1991. Print.

[12] Chabot, J. and Johnson, M. and Kelley, J. and Thein, M. L., "Using TableSat IC for the Analysis of Attitude Control and Flexible Boom Dynamics for NASA Magnetospheric MultiScale (MMS) Mission Spacecraft," Proceedings of the Proceedings of the 23rd AAS/AIAA Space Flight Mechanics Meeting, AAS 13-291, Kauai, HI, February 2013.

[13] "Arduino - ArduinoBoardMega2560." Arduino - ArduinoBoardMega2560. N.p., n.d. Web. 15 Aug. 2016.

[14] XBee®/XBee-PRO® RF Modules. N.p.: Digi International, Inc., 2009. PDF.

[15] "9 Degrees of Freedom - Razor IMU." Sparkfun. N.p., n.d. Web. 15 Aug. 2016.

[16] Badger, Paul. "Arduino Playground - CapacitiveSensor." Arduino Playground - CapacitiveSensor. N.p., n.d. Web. 15 Aug. 2016.

[17] OMEGA. Precision Strain Gauge. N.p.: OMEGA, 2016. PDF.

[18] Texas Instruments. Instrumentation Amplifier INA125. N.p.: Texas Instruments, 2016. PDF.

[19] Fenner, Patrick. "Reading Strain Gauge Scales with Arduino." Reading Strain Gauge Scales with Arduino. N.p., 14 June 2013. Web. 15 Aug. 2016.

[20] "Magnetospheric Multiscale: Using Earth's magnetosphere as a laboratory to study the microphysics of magnetic reconnection" PDF. NASA. March 2015.

[21] Lindqvist, P., Olsson, G., Torbert, R.B. et al. Space Sci Rev (2016) 199: 137.

Appendices

Appendix A

Finite Element Model Global Mass, Spring, and Damping Matrices

This appendix contains the full global mass spring and damping matrices for the FEM.

$$M = \begin{bmatrix} 109.8215 & 0 & 0 & 0 & 0 & 27.4554 & 0 & 0 & 0 & 0 \\ 0122.3725 & 0 & 0 & 0 & 0 & 21.1799 & 0 & 0 & -1.0959 & \\ 0 & 0 & 122.3725 & 0 & 0 & 0 & 0 & 21.1799 & 1.0959 & 0 \\ 0 & 0 & 0 & 0.1449 & 0 & 0 & 0 & -1.0959 & -0.0544 & 0 \\ 0 & 0 & 0 & 0 & 0.1449 & 0 & 1.0959 & 0 & 0 & -0.0544 \\ 27.4554 & 0 & 0 & 0 & 0 & 54.9108 & 0 & 0 & 0 & 0 \\ 0 & 21.1799 & 0 & 0 & 1.0959 & 0 & 61.1863 & 0 & 0 & -1.8546 \\ 0 & 0 & 21.1799 & -1.0959 & 0 & 0 & 0 & 61.1863 & 1.8546 & 0 \\ 0 & 0 & 1.0959 & -0.0544 & 0 & 0 & 0 & 1.8546 & 0.0725 & 0 \\ 0 & -1.0959 & 0 & 0 & -0.0544 & 0 & -1.8546 & 0 & 0 & 0.0725 \end{bmatrix} \times 10^{-5} \tag{A.1}$$

$$K = \begin{bmatrix} 1.78 \times 10^6 & 0 & 0 & 0 & 0 & -8.92 \times 10^5 & 0 & 0 & 0 & 0 \\ 0 & 87.86 & 0 & 0 & 0 & 0 & -43.93 & 0 & 0 & 4.72 \\ 0 & 0 & 15.56 & 0 & 0 & 0 & 0 & -7.78 & -0.84 & 0 \\ 0 & 0 & 0 & 0.24 & 0 & 0 & 0 & 0.84 & 0.06 & 0 \\ 0 & 0 & 0 & 0 & 1.35 & 0 & -4.72 & 0 & 0 & 0.34 \\ -8.92 \times 10^5 & 0 & 0 & 0 & 0 & 8.92 \times 10^5 & 0 & 0 & 0 & 0 \\ 0 & -43.93 & 0 & 0 & -4.72 & 0 & 43.93 & 0 & 0 & -4.72 \\ 0 & 0 & -7.78 & 0.84 & 0 & 0 & 0 & 7.78 & 0.84 & 0 \\ 0 & 0 & -0.84 & 0.06 & 0 & 0 & 0 & 0.84 & 0.12 & 0 \\ 0 & 4.72 & 0 & 0 & 0.34 & 0 & -4.72 & 0 & 0 & 0.68 \end{bmatrix} \tag{A.2}$$

$$B = \begin{bmatrix} 5.8604 & 6.1534 & 6.1534 & 0 & 0 & 1.4651 & 1.3186 & 1.3186 & -0.0630 & -0.0630 \\ 6.1534 & 6.5302 & 6.5302 & 0 & 0 & 1.3186 & 1.1302 & 1.1302 & -0.0585 & -0.0585 \\ 6.1534 & 6.5302 & 6.5302 & 0 & 0 & 1.3186 & 1.1302 & 1.1302 & -0.0585 & -0.0585 \\ 0 & 0 & 0 & 0.0077 & 0.0077 & 0.0630 & 0.0585 & 0.0585 & -0.0029 & -0.0029 \\ 0 & 0 & 0 & 0.0077 & 0.0077 & 0.0630 & 0.0585 & 0.0585 & -0.0029 & -0.0029 \\ 1.4651 & 1.3186 & 1.3186 & 0.0630 & 0.0630 & 2.9302 & 3.0767 & 3.0767 & -0.0945 & -0.0945 \\ 1.3186 & 1.1302 & 1.1302 & 0.0585 & 0.0585 & 3.0767 & 3.2651 & 3.2651 & -0.0990 & -0.0990 \\ 1.3186 & 1.1302 & 1.1302 & 0.0585 & 0.0585 & 3.0767 & 3.2651 & 3.2651 & -0.0990 & -0.0990 \\ -0.0630 & -0.0585 & -0.0585 & -0.0029 & -0.0029 & -0.0945 & -0.0990 & -0.0990 & 0.0039 & 0.0039 \\ -0.0630 & -0.0585 & -0.0585 & -0.0029 & -0.0029 & -0.0945 & -0.0990 & -0.0990 & 0.0039 & 0.0039 \end{bmatrix} \times 10^{-3} \qquad \text{(A.3)}$$

Appendix B

Wiring Diagrams and Sensor Code

This appendix contains the wiring diagrams and Arduino code for the accelerometers, capacitive displacement sensor, and strain gauges.

B.1 Accelerometers

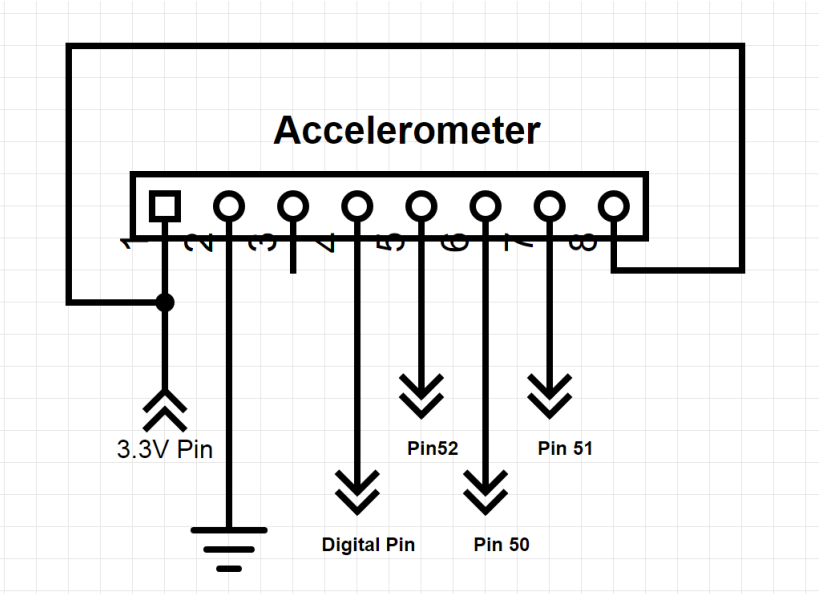

Figure B.1: Wiring diagram for the accelerometers

```
// Accelerometer Code

#include <SPI.h>
#include "bma180.h"

/*
Arduino Mega SPI pins are 50 MISO, 51 MOSI, 52 SCK,
53 SS
*/

#define bma_pin_11 22      //bma selection pin 1
#define bma_pin_12 23      //bma selection pin 2
#define bma_pin_10 24      //bma selection pin 3
#define bma_pin_1 25      //bma selection pin 4
#define bma_pin_2 26      //bma selection pin 5
#define bma_pin_3 27      //bma selection pin 6
#define bma_pin_7 28      //bma selection pin 7
#define bma_pin_9 29      //bma selection pin 8
#define bma_pin_5 30      //bma selection pin 9
#define bma_pin_8 31      //bma selection pin 10
#define bma_pin_4 32      //bma selection pin 11
#define bma_pin_6 33      //bma selection pin 12
#define bmas_connected 12 //change this number depending on how many
//are connected
bma180 bmas[bmas_connected];
int pins[bmas_connected] = {
bma_pin_1,
bma_pin_2,
bma_pin_3,
bma_pin_4,
bma_pin_5,
bma_pin_6,
bma_pin_7,
bma_pin_8,
bma_pin_9,
bma_pin_10,
bma_pin_11,
bma_pin_12
};

void setup()
{
Serial.begin(9600);
pinMode(53, OUTPUT); //the SPI documentation says that
//this pin must be put into output mode or SPI won't work,
//even if its unused.
pinMode(bma_pin_1, OUTPUT);
pinMode(bma_pin_2, OUTPUT);
```

```
pinMode(bma_pin_3, OUTPUT);
pinMode(bma_pin_4, OUTPUT);
pinMode(bma_pin_5, OUTPUT);
pinMode(bma_pin_6, OUTPUT);
pinMode(bma_pin_7, OUTPUT);
pinMode(bma_pin_8, OUTPUT);
pinMode(bma_pin_9, OUTPUT);
pinMode(bma_pin_10, OUTPUT);
pinMode(bma_pin_11, OUTPUT);
pinMode(bma_pin_12, OUTPUT);

for(int i = 0; i < bmas_connected; i++)
{
digitalWrite(pins[i], HIGH);
}

SPI.begin();
SPI.setBitOrder(MSBFIRST); //big endian
SPI.setDataMode(SPI_MODE0); //document syas this is 2 but I
//tried and it only worked correctly on 0

for(int i = 0; i < bmas_connected; i++)
{
digitalWrite(pins[i], LOW);
bmas[i].init();
digitalWrite(pins[i], HIGH);
}

delay(50);
}

void loop()
{
for(int i = 0; i < bmas_connected; i++)
{
digitalWrite(pins[i], LOW);
Serial.print(millis());
Serial.print(",");
Serial.print(i);
Serial.print(",");
bmas[i].readAccel();
digitalWrite(pins[i], HIGH);
}
}
```

B.2 Capacitive Displacement Sensor

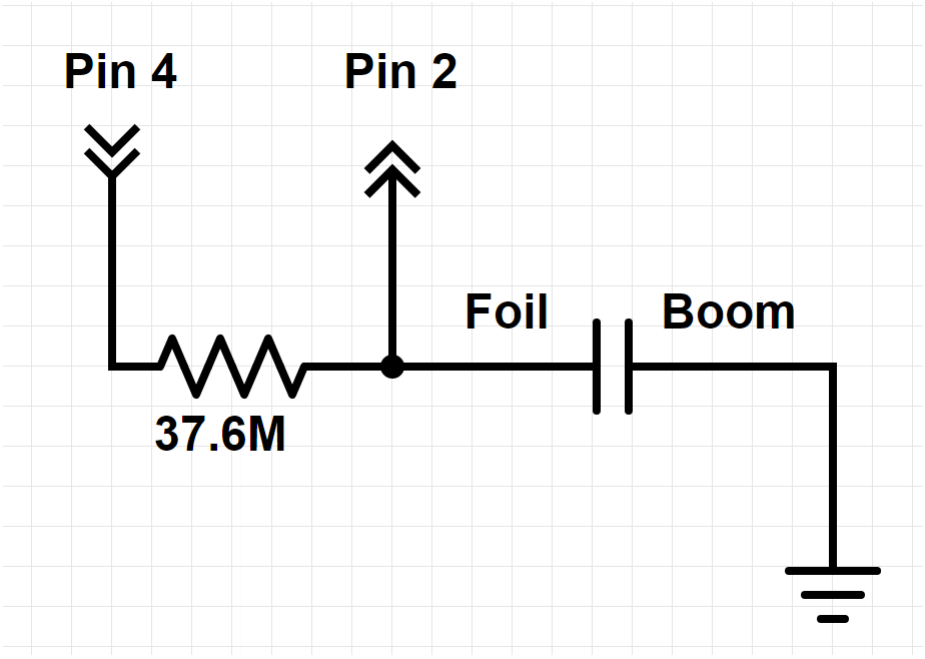

Figure B.2: Wiring diagram for the capacitive displacement sensor

```
#include <CapacitiveSensor.h>

/*
* CapitiveSense Library Demo Sketch
* Paul Badger 2008
* Uses a high value resistor e.g. 10 megohm between
* send pin and receive pin
* Resistor effects sensitivity, experiment with values,
* 50 kilohm - 50 megohm.
* Larger resistor values yield larger sensor values.
* Receive pin is the sensor pin -
* try different amounts of foil/metal on this pin
* Best results are obtained if sensor foil and wire is covered with an
* insulator such as paper or plastic sheet
*/


CapacitiveSensor   cs_4_2 = CapacitiveSensor(4,2);
// 10 megohm resistor between pins 4 & 2,
// pin 2 is sensor pin, add wire, foil
CapacitiveSensor   cs_4_5 = CapacitiveSensor(4,5);
// 10 megohm resistor between pins 4 & 6,
// pin 6 is sensor pin, add wire, foil
CapacitiveSensor   cs_4_8 = CapacitiveSensor(4,8);
// 10 megohm resistor between pins 4 & 8,
```

```
// pin 8 is sensor pin, add wire, foil

void setup()
{
cs_4_2.set_CS_AutocaL_Millis(0xFFFFFFFF);
// turn off autocalibrate on channel 1 - just as an example
Serial.begin(9600);
}

void loop()
{
long start = millis();
long total1 =  cs_4_2.capacitiveSensor(30);
long total2 =  cs_4_5.capacitiveSensor(30);
long total3 =  cs_4_8.capacitiveSensor(30);

Serial.print(millis() - start);
// check on performance in milliseconds
Serial.print("\t");
// tab character for debug window spacing

Serial.print(total1);                  // print sensor output 1
Serial.print("\t");
Serial.print(total2);                  // print sensor output 2
Serial.print("\t");
Serial.println(total3);                // print sensor output 3

delay(10);
// arbitrary delay to limit data to serial port
}
```

B.3 Strain Gauges

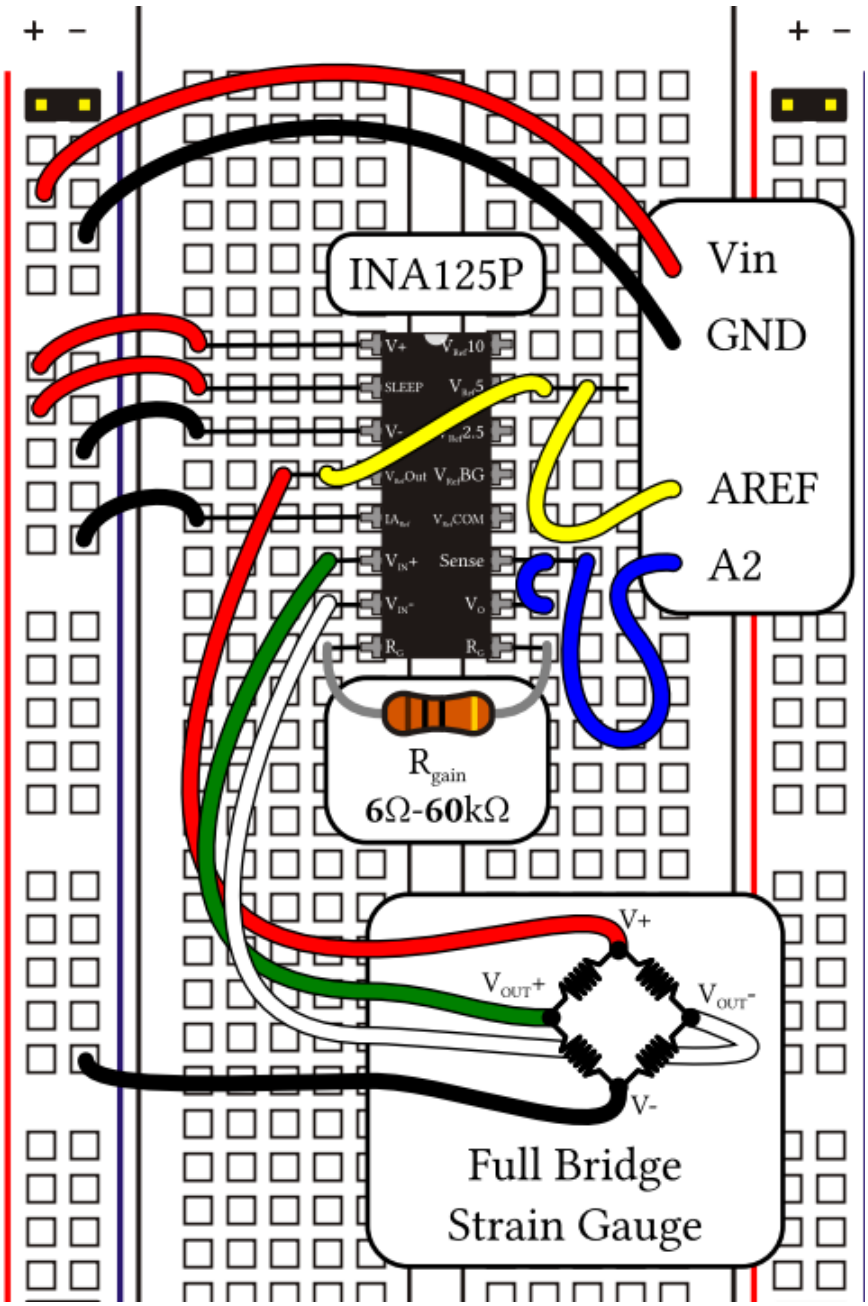

Figure B.3: Wiring diagram for the strain gauge sensor

```
/*
Strain Gauge Code
*/

// Global Variables
const int analogPinPos = A2;
const int analogPinNeg = A0;
int sensorValuePos = 3;
```

```
int sensorValueNeg = 3;
unsigned long time;

void setup() {
// Start the
Serial.begin(57600);
delay(500);
Serial.println("Arduino Weight Measurement");
Serial.println("=========================");

// Set an analogue voltage reference of the return from the V_{in} pin.
// This means you use the full range of the Atmega's A2D converter.
analogReference(EXTERNAL);
}

void loop() {
// Read the amplified output from the strain gauges.
int sensorValuePos = analogRead(analogPinPos);
int sensorValueNeg = analogRead(analogPinNeg);
time = millis();

// Delayed serial output that doesn't interfere with the gauge refresh rate

Serial.print(time);
Serial.print(" ");

//    Serial.print(":sensorValuePos: ");
//    Serial.print(sensorValuePos);

Serial.print(":sensorValueNeg: ");
Serial.print(sensorValueNeg);

Serial.println();
}
```

Appendix C

Modified Hybrid Algorithm Code

This appendix contains the code for the modified hybrid algorithm.

```
%Christopher Hashem
%Modified Hybrid Algorithm
%January 2014 - September 2016

clear
close all
clc

% Parameters
RPY(:,1)=[0 0 0];
w(:,1)=[0 0 0];
alpha(:,1)=[0 0 0];
desired_rate=[0;0;3];       %% desired angular velocity of hub [rad/s]
timestep=.01;               %% system timestep [s]
deltat=.0001;               %% FEA time step [s]
r_hub=.1518;                %% hub radius [m]
h_hub=.10795;               %% hub height [m]
m_hub=3;                    %% hub mass [kg]
E=2E11;%69E9;               %% boom elastic modulus [Pa]
rho=8000;%2700;             %% boom material density [kg/m^3]
height=1.50876E-3;          %% boom height [m]
width=6.35E-4;              %% boom width [m]
A=width*height;             %% boom cross sectional area [m^2]
IAzz=width*height^3/12;     %% area moment about y [m^4]
IAyy=height*width^3/12;     %% area moment about z [m^4]
L=.4298578;                 %% boom length [m]
l=L/2;                      %% boom element length [m]
mu=1*.0409;                 %% damping coefficient of boom
mu_drag=1.846E-5;           %% dynamic viscosity of air
gamma=1/2;                  %% Newmark-beta constant
beta=1/4;                   %% Newmark-beta constant
thuster=.3111;              %% thruster saturation N [F]
LQR=0;                      %% set to 1 if LQR, 0 if PID/SMC
```

```
syms x
N=[1-x/l (2*x^3-3*l*x^2+l^3)/l^3 (2*x^3-3*l*x^2+l^3)/l^3 ...
(l*x^3-2*l^2*x^2+l^3*x)/l^3 (l*x^3-2*l^2*x^2+l^3*x)/l^3 ...
x/l (-2*x^3+3*l*x^2)/l^3 (-2*x^3+3*l*x^2)/l^3 ...
(l*x^3-l^2*x^2)/l^3 (l*x^3-l^2*x^2)/l^3];  %% shape function
N2=transpose(N)*N;

IM_hub=[m_hub*(h_hub^2+3*r_hub^2)/12;...
m_hub*(h_hub^2+3*r_hub^2)/12;...
(m_hub*r_hub^2)/2];     %% hub mass moment of inertia
Ix_thinplate=rho*A*l*(height^2+width^2)/12;
Iy_thinplate=rho*A*l*(l^2+width^2)/12;
Iz_thinplate=rho*A*l*(height^2+l^2)/12;
IM_boom(:,1)=...
[2*(Ix_thinplate+Iz_thinplate+2*rho*A*l*(r_hub+l)^2);...
2*(Ix_thinplate+Iz_thinplate+2*rho*A*l*(r_hub+l)^2);...
4*(Iy_thinplate+2*rho*A*l*(r_hub+l)^2)];%% boom mass moment
IM_system(:,1)=IM_hub+IM_boom(:,1);   %% system mass moment
Mrxn(:,1)=[0;0;0];          %% reaction moment
Mdrag(:,1)=[0;0;2*pi*mu_drag*w(3,1)*r_hub^2*h_hub+...
1.9*A*1.2041*w(3,1)*((r_hub+L)^3-r_hub^3)/6];  %% drag
error(:,1)=[0;0;3];         %% error initialization

ma=[70 0 0 0 0;...
0 78 0 0 11*l;...
0 0 78 -11*l 0;...
0 0 -11*l 2*l^2 0;...
0 11*l 0 0 2*l^2];

mb=[35 0 0 0 0;...
0 27 0 0 -6.5*l;...
0 0 27 6.5*l 0;...
0 0 -6.5*l -1.5*l^2 0;...
0 6.5*l 0 0 -1.5*l^2];

mc=[35 0 0 0 0;...
0 27 0 0 6.5*l;...
0 0 27 -6.5*l 0;...
0 0 6.5*l -1.5*l^2 0;...
0 -6.5*l 0 0 -1.5*l^2];

md=[70 0 0 0 0;...
0 78 0 0 -11*l;...
0 0 78 11*l 0;...
0 0 11*l 2*l^2 0;...
0 -11*l 0 0 2*l^2];

M=((rho*l*A)/210)*...
```

```
[ma+md mb;...
mc md];                  %%FEA mass matrix

b=mu*double(int(N2,x,0,l));
ba=b(1:5,1:5);
bb=b(1:5,6:10);
bc=b(6:10,1:5);
bd=b(6:10,6:10);
B=[bd+ba bb
bc bd];                  %%FEA damping matrix

ka=[E*A/l 0 0 0 0;...
0 12*E*IAzz/l^3 0 0 6*E*IAzz/l^2;...
0 0 12*E*IAyy/l^3 -6*E*IAyy/l^2 0;...
0 0 -6*E*IAyy/l^2 4*E*IAyy/l 0;...
0 6*E*IAzz/l^2 0 0 4*E*IAzz/l];

kb=[-E*A/l 0 0 0 0;...
0 -12*E*IAzz/l^3 0 0 6*E*IAzz/l^2;...
0 0 -12*E*IAyy/l^3 -6*E*IAyy/l^2 0;...
0 0 6*E*IAyy/l^2 2*E*IAyy/l 0;...
0 -6*E*IAzz/l^2 0 0 2*E*IAzz/l];

kc=[-E*A/l 0 0 0 0;...
0 -12*E*IAzz/l^3 0 0 -6*E*IAzz/l^2;...
0 0 -12*E*IAyy/l^3 6*E*IAyy/l^2 0;...
0 0 -6*E*IAyy/l^2 2*E*IAyy/l 0;...
0 6*E*IAzz/l^2 0 0 2*E*IAzz/l];

kd=[E*A/l 0 0 0 0;...
0 12*E*IAzz/l^3 0 0 -6*E*IAzz/l^2;...
0 0 12*E*IAyy/l^3 6*E*IAyy/l^2 0;...
0 0 6*E*IAyy/l^2 4*E*IAyy/l 0;...
0 -6*E*IAzz/l^2 0 0 4*E*IAzz/l];

K=[ka+kd kb;...
kc kd];                  %%FEA stiffness matrix

a0=1/(beta*deltat^2);       %%FEA coefficients
a1=gamma/(beta*deltat);
a2=1/(beta*deltat);
a3=1/(2*beta)-1;
a4=gamma/beta -1;
a5=deltat*(gamma/beta-2)/2;
a6=deltat*(1-gamma);
a7=deltat*gamma;

Keff=K+a0*M+a1*B;           %% effective stiffness matrix
```

```
% FEA Initial Conditions
f(:,1)=[0;0;0;0;0;0;0;0;0;0];
u(:,1)=[0;0;0;0;0;0;0;0;0;0];
u_dot(:,1)=[0;0;0;0;0;0;0;0;0;0];
u_dot_dot(:,1)=[0;0;0;0;0;0;0;0;0;0];

%% Algorithm

for j=2:500
%% Rigid Body Dynamics
j%=2;
if j>=150 && j<=200
disturbance=0;%-.5;
else
if j>=300 && j<=400
disturbance=0;%-.225;
else
disturbance=0;
end
end
RPY_0=RPY(:,j-1);
w_0=w(:,j-1);
IM_system(:,j)=IM_hub+IM_boom(:,j-1);
if LQR==1
A_ss=[0 -desired_rate(3)*(IM_system(3,j)-IM_system(2,j))/IM_system(1,j) 0;...
-desired_rate(3)*(IM_system(1,j)-IM_system(3,j))/IM_system(2,j) 0 0;...
0 0 0];
B_ss=[1/IM_system(1,j) 0 0;...
0 1/IM_system(2,j) 0;...
0 0 1/IM_system(3,j)];
C_ss=eye(3);
D_ss=zeros(3);
Q=diag([1 1 1]);
R=diag([1 1 1]);
[Klqr,S,e]=lqr(A_ss,B_ss,Q,R);
else
end
sim('RigidBodyDynamics')
RPY(:,j)=transpose(RPYsim(2,:));
w(:,j)=transpose(wsim(2,:));
alpha(:,j)=transpose(alphasim(1,:));
effort(:,j)=transpose(effortsim(1,:));
thruster(:,j)=transpose(thrustersim(1,:));
error(:,j)=transpose(errorsim(1,:));
Mdrag(:,j)=[0;0;2*pi*mu_drag*w(3,j)*r_hub^2*h_hub+...
1.9*A*1.2041*w(3,j)*((r_hub+L)^3-r_hub^3)/6]*1.9E5;
```

```
%% Boundary Conditions
chord1=2*(r_hub+l)*sin((RPY(:,j)-RPY(:,j-1))/2);
chord2=2*(r_hub+2*l)*sin((RPY(:,j)-RPY(:,j-1))/2);
for k=1:3
if RPY(k,j)==0
ABG(k)=0;
else
ABG(k)=asin((r_hub+l)*sin((RPY(k,j)-RPY(k,j-1)))/chord1(k));
end
end
Pbc=[-chord1(2).*cos(ABG(2))-chord1(3).*cos(ABG(3));...
chord1(2).*sin(ABG(2));...
chord1(3).*sin(ABG(3));...
0;0;...
-chord2(2).*cos(ABG(2))-chord2(3).*cos(ABG(3));...
chord2(2).*sin(ABG(2));...
chord2(3).*sin(ABG(3));...
0;0];
Vbc=[-chord1(2).*cos(ABG(2))-chord1(3).*cos(ABG(3));...
chord1(2).*sin(ABG(2));...
chord1(3).*sin(ABG(3));0;0;...
-chord2(2).*cos(ABG(2))-chord2(3).*cos(ABG(3));...
chord2(2).*sin(ABG(2));chord2(3).*sin(ABG(3));0;0]/timestep;
U(:,1)=u(:,j-1)+Pbc;
U_dot(:,1)=u_dot(:,j-1)+0*Vbc;
U_dot_dot(:,1)=u_dot_dot(:,j-1);
%% Finite Element Analysis
for i=2:100

f(:,i)=[0;0;0;0;0;0;0;0;0;0];
if j==200
f(:,2)=[0;0;0;0;0;0;0;0;0;0];
else
f(:,i)=[0;0;0;0;0;0;0;0;0;0];
end

Feff(:,i)=f(:,i)+M*(a0*U(:,i-1)...
+a2*U_dot(:,i-1)+a3*U_dot_dot(:,i-1))+...
B*(a1*U(:,i-1)+a4*U_dot(:,i-1)+a5*U_dot_dot(:,i-1));

U(:,i)=inv(Keff)*Feff(:,i);

U_dot_dot(:,i)=a0*(U(:,i)-U(:,i-1))...
-a2*U_dot(:,i-1)-a3*U_dot_dot(:,i-1);

U_dot(:,i)=U_dot(:,i-1)+a6*U_dot_dot(:,i-1)+a7*U_dot_dot(:,i);
end
u(:,j)=U(:,i);
```

```
u_dot(:,j)=U_dot(:,i);
u_dot_dot(:,j)=U_dot_dot(:,i);
%% Boom Inertia Calculation
if u(:,j)==[0;0;0;0;0;0;0;0;0;0]
IM_boom(:,j)=IM_boom(:,j-1);
else

x2=u(1,j);
y2=u(2,j);
z2=u(3,j);
x3=u(6,j);
y3=u(7,j);
z3=u(8,j);
if z2<.0001
x2=0;
else
A1=.5*z2*l;
theta1=z2/l;
f1=(2*A1)/(l*sin(theta1));
g1=sqrt(f1^2+l^2-2*f1*l*cos(theta1));
x2=sqrt(g1^2-z2^2);
end
if z3<.0001
x3=0;
else
l2=2*l-x2;
A2=.5*z3*l2;
theta2=z3/l2;
f2=(2*A2)/(l2*sin(theta2));
g2=sqrt(f2^2+l2^2-2*f2*l2*cos(theta2));
x2=sqrt(g2^2-z3^2);
end

%xOy Plane
I_xOy=A*l*rho*((z2/2)^2+((z3+z2)/2)^2);

%yOz Plane
I_yOz=A*l*rho*(((l-x2)/2)^2+((-x3-x2+2*l)/2)^2);

%zOx Plane
I_zOx=A*l*rho*((y2/2)^2+((y3+y2)/2)^2);

IMbar_xx=I_xOy+I_zOx;
IMbar_yy=I_xOy+I_yOz;
IMbar_zz=I_yOz+I_zOx;

%%% Parallel Axis Theorem
IMxx=2*(IMbar_xx+IMbar_zz+2*rho*A*l*r_hub^2);
```

```
IMyy=2*(IMbar_xx+IMbar_zz+2*rho*A*l*r_hub^2);
IMzz=4*(IMbar_yy+2*rho*A*l*r_hub^2);

%%% Coordinate Transformation
IM_boom(:,j)=[IMxx;IMyy;IMzz];
end
%% Reaction Torque Calculation
if u(:,j)==[0;0;0;0;0;0;0;0;0;0]
Mrxn(:,j)=[0;0;0];
else
Frxn_z=3*E*IAyy*z3/L^3;
Mrxn(:,j)=(r_hub+L)*[0;0;4*Frxn_z];
end

end

%% Post Processing
figure
plot(w(3,:),'b')
hold on
plot(w(2,:),'r')
plot(w(1,:),'c')
title('Yaw Rate of Rigid Body')
xlabel('Iteration')
ylabel('Rate of Rotation (rad/s)')
legend('Yaw rate')
figure
plot(u(1,:),'b')
hold on
plot(u(2,:),'r')
plot(u(3,:),'m')
title('Node 2 Displacement')
xlabel('Iteration')
ylabel('Displacement (m)')
legend('u2','v2','w2')
figure
plot(u(6,:),'b')
hold on
plot(u(7,:),'m')
plot(u(8,:),'r')
title('Node 3 Displacement')
xlabel('Iteration')
ylabel('Displacement (m)')
legend('u3','v3','w3')
figure
plot(IM_system(1,:),'b')
hold on
plot(IM_system(2,:),'r')
```

```
plot(IM_system(3,:),'k')
title('System Mass Moment of Inertia')
xlabel('Iteration')
ylabel('Mass Moment of Inertia (kg*m^2)')
legend('Ixx','Iyy','Izz')
figure
plot(effort(3,:))
title('Control Effort')
xlabel('Iteration')
ylabel('Effort (N)')
figure
plot(thruster(3,:))
title('Thruster Output')
xlabel('Iteration')
ylabel('Force (N)')
```